VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

# DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Aleš Musil


# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS


# BEZDRÁTOVÝ MĚŘICÍ SYSTÉM S BATERIOVÝM NAPÁJENÍM
BATTERY POWERED WIRELESS MEASURING SYSTEM

**DIPLOMOVÁ PRÁCE**
MASTER'S THESIS


**AUTOR PRÁCE** — **Bc. Aleš Musil**
AUTHOR

**VEDOUCÍ PRÁCE** — **Ing. Ondřej Krajsa, Ph.D.**
SUPERVISOR


**BRNO 2020**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**

Ústav telekomunikací

***Student:*** Bc. Aleš Musil ***ID:*** 174230
***Ročník:*** 2 ***Akademický rok:*** 2019/20

**NÁZEV TÉMATU:**

## Bezdrátový měřicí systém s bateriovým napájením

**POKYNY PRO VYPRACOVÁNÍ:**

Navrhněte a realizujte bezdrátový senzorový systém, skládající se z brány a vzdálené jednotky. Vzdálená jednotka bude napájena z baterie s případným dobíjením ze solárního panelu. Pro demonstrační funkci bude plnit úlohu měření teploty, tlaku a vlhkosti vzduchu. Brána bude odesílat naměřená data pomocí sítě Internet.

**DOPORUČENÁ LITERATURA:**

[1] KYUNG, Chong-Min, Hiroto YASUURA, Yongpan LIU a Youn-Long LIN. Smart Sensors and Systems: Innovations for Medical, Environmental, and IoT Applications. Cham: Springer, 2016. DOI: 10.1007/978-3-3-9-33201-7. ISBN 9783319332000.

[2] MACII, Enrico. Ultra low-power electronics and design. Dordrecht: Kluwer Academic Publishers, 2004, 273 s. ISBN 1-4020-8075-1.

***Termín zadání:*** 3.2.2020 ***Termín odevzdání:*** 1.6.2020

***Vedoucí práce:*** Ing. Ondřej Krajsa, Ph.D.

**prof. Ing. Jiří Mišurec, CSc.**
předseda oborové rady

## ABSTRAKT

Diplomová práce se zabývá návrhem a realizací bezdrátového měřícího systému s bateriovým napájením. Systém je složen ze dvou částí. První, vzdálená jednotka, je navržena tak aby dosahovala co nejmenší spotřeby, k tomu využívá mikrokontrolér z rodiny STM32. Data jsou odesílána pomocí RFM69. Brána běží na operačním systému Linux za použití Raspberry Pi. Data jsou z hlavního programu brány odeslána pomocí MQTT a zobrazena za pomoci projektu Home Assistant.



## KLÍČOVÁ SLOVA

Bezdrátový měřící systém, MQTT, Raspberry Pi, Linux, RFM69, STM32L031K6, Rust, C, Docker, Kontejner



## ABSTRACT

Master thesis deals with concept and realization of wireless measuring system, powered by battery. The system consists of two main parts. First part, the node, is designed for the lowest possible power consumption. This is achieved mainly by usage of microcontroller from the STM32 family. Gathered information is transfered via RFM69 module. Second part, the gateway, is powered by Linux operating system on Raspberry Pi. Data from the gateway proxy are propagated through MQTT into Home Assistant, which interprets the results.



## KEYWORDS

Wireless measuring system, MQTT, Raspberry Pi, Linux, RFM69, STM32L031K6, Rust, C, Docker, Container

MUSIL, Aleš. *Bezdrátový měřicí systém s bateriovým napájením*. Brno, 2020, 73 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.

# PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Bezdrátový měřicí systém s bateriovým napájením“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ..... .....

podpis autora

# PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Ondřeji Krajsovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

# Obsah

# Seznam obrázků

# Seznam tabulek

# Seznam výpisů

# 1 Úvod

Práce se zabývá bezdrátovým měřícím systémem, který je napájen pomocí baterie. V současnosti je u všech zařízení kladen důraz na co největší výdrž a spolehlivost systému při uchování co nejmenších rozměrů. Součástí práce je vytvoření dvou částí měřícího systému a vzdálené jednotky, která je napájena baterií a přídavným solárním panelem, a dále bránou, která obstarává zobrazování a sběr informací.

V první části práce jsou rozebrány teoretické informace k problematice, teoretický návrh samotného měřícího systému a jednotlivých částí. Dále obsahuje úvahu nad možnostmi různých řešení a nakonec výběr toho nejvhodnějšího. S ohledem na velké množství bezdrátových technologií, které jsou dostupné, je popsáno několik hojně využívaných jako je ZigBee, Bluetooth nebo Wi-Fi. Na konci kapitoly je uvedeno porovnání a výběr technologie, která bude použita pro realizaci vzdálené jednotky.

Mezi další popsané technologie, které jsou použity v rámci práce, patří například ARM, který je velmi využíván a v posledních letech se nachází téměř v každém zařízení nehledě na funkci a velikost. Procesory nebo mikrokontroléry s touto technologií se vyznačují velmi dobrými parametry v oblasti, na kterou jsou specializovány.

V domácí automatizaci nebo IoT se velmi hojně využívá technologie MQTT, jejíž základní funkční popis se nachází v teoretické části práce. MQTT v kombinaci s projektem Home Assistant vytváří skvělou platformu, pomocí které lze nejen sbírat data, ale také ovládat různá zařízení za pomoci internetového prohlížeče nebo mobilní aplikace.

V druhé části práce jsou rozebrány již praktické postupy, za pomoci kterých bylo docíleno zadaného úkolu. V první řadě je vysvětlen komunikační protokol mezi vzdálenou jednotkou a bránou se všemi vrstvami. Dále je popsána samotná vzdálená jednotka s návrhem DPS a výběr vhodných komponent pro dodržení co nejmenší spotřeby, resp. co největší životnosti baterie. V neposlední řadě je také popsán firmware, který běží na vzdálené jednotce.

Následuje výběr vhodného HW pro bránu, aby byl splněn požadavek možnosti komunikovat přes síť Internet a zároveň dostatečně výkonný pro běh již zmiňovaného Home Assistanta. Je popsán software, který se stará o komunikaci se vzdálenou jednotkou a komunikací s Home Assistantem pomocí MQTT.

Poslední kapitola obsahuje naměřená data o výsledné vzdálené jednotce, odběr jednotky a předpokládanou výdrž při použití určitého druhu baterie, a nakonec dosah signálu vzdálené jednotky.

# 2 Návrh systému

Bezdrátový měřící systém je složen ze dvou základních částí, které jsou nezbytné pro fungování výsledného produktu. Specifické požadavky jsou kladeny především na vzdálenou jednotku. Přístup k naměřeným datům je možný pomocí sítě Internet.

S tímto faktem je spojen pojem IoT (Internet of Things), jedná se o technologii, která je v posledních letech na vzestupu. Tato technologie umožňuje interakci mezi výpočetními zařízeními, které lze unikátně rozlišit, a jinými stroji nebo lidmi. To vše za účelem sběru dat a automatizace určitých úkonů po drátové nebo bezdrátové síti Internet [1].

## 2.1 Brána

Jak již bylo uvedeno, brána musí být schopna komunikovat pomocí sítě Internet, protože slouží jako brána pro přístup k naměřeným datům. Druhý a zároveň poslední požadavek na bránu je schopnost komunikace s jednotkou vzdálenou. Tyto dva požadavky dávají k dispozici poměrně velký počet možností dostupných řešení. Zjednodušené blokové schéma brány lze vidět na obrázku 2.1. Problematiku výběru řešení lze rozdělit na tři základní možnosti:

- Vlastní jednoúčelové řešení
- Existující jednoúčelové řešení
- Víceúčelové řešení s operačním systémem

### 2.1.1 Vlastní jednoúčelové řešení

Jak již název napovídá, jedná se v podstatě o řešení vyrobené na míru danému problému. V tomto případě musí splňovat specifické požadavky, kterých není mnoho. Výhodou vlastního řešení je úplná kontrola nad specifikacemi výsledného produktu, především spotřeba energie, výkon systému, cena. Nevýhod je hned několik. V první řadě vzrůstá komplexnost celého řešení, protože je nutné vzít v úvahu úkony spojené s návrhem produktu. To znamená výběr správných HW (Hardware) komponent, ze kterých je následně vytvořeno schéma zapojení, DPS (Deska plošných spojů) a osazení. Poté je nutné vytvořit vhodný SW (Software) a udržovat jej v chodu. Z pohledu využitelnosti převažují spíše negativní aspekty než pozitivní, proto nebyl zvolen tento přístup.

### 2.1.2 Existující jednoúčelové řešení

Eliminuje téměř všechny nevýhody vlastního jednoúčelového řešení, avšak vnáší do systému zcela jiný problém, tzv. vendor lock-in neboli proprietární uzavření. Jedná

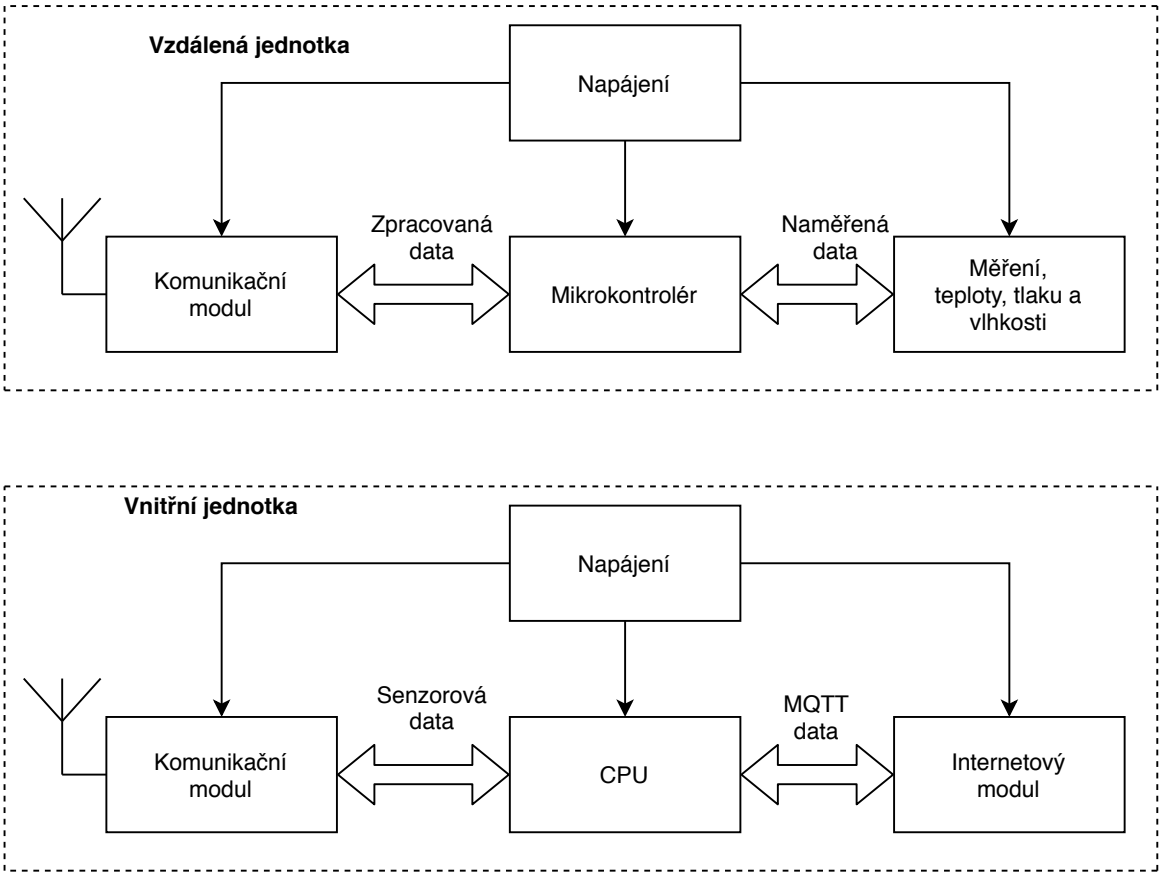
se o stav, kdy je jedinec vázán na danou technologii a přechod na jiný systém by se mohl stát velice nákladným a nevýhodným [2]. K uzavření by mohlo dojít v obou stěžejních částech, tedy jak pro HW, tak i SW. Tento problém hraje důležitou roli při výběru konečného řešení, z tohoto důvod nebyl zvolen ani tento přístup.

### 2.1.3 Víceúčelové řešení s operačním systémem

Využití operačního systému přidává obrovské možnosti především z pohledu SW. Výhodou je možnost najít existující řešení, které již někdo vytvořil, a použít, popřípadě poskládat z dostupných částí, především SW knihoven. Další nespornou výhodou je přítomnost ovladačů na obrovské množství zařízení přímo v jádře operačního systému. Při použití vhodného HW naprosto odpadá nutnost řešení připojení do sítě Internet. Nevýhodou je větší cena dostatečně výkonného HW, které je schopné běžet operační systém.

## 2.2 Vzdálená jednotka

Jednotka provádí periodický sběr určených dat, která jsou odeslána do brány. Je požadováno, aby byla vzdálená jednotka napájena baterií s případným dobíjením ze solárních článků. Z tohoto požadavku vyplývá nutnost co nejmenší energetické náročnosti, čímž se prodlouží doba běhu na baterii. Jednotka je složena ze čtyř hlavních komponent, jak lze vidět na obrázku 2.1. Je nutné, aby všechny komponenty měly režim nízké spotřeby, v případě napájecího obvodu je důležitý především klidový proud. U mikrokontroléru je zároveň výhodné využít režimů vnitřního regulátoru, které umožňují mnohem nižší spotřebu [3].




Obr. 2.1: Blokové schéma systému

# 3 Bezdrátové senzorové sítě

Bezdrátové senzorové sítě jsou rozsáhlé distribuované systémy, které sbírají a zpracovávají data z mnoha oblastí. Jsou většinou složeny z mnoha jednotek rozmístěných napříč různým prostředím. Přestože samotná jednotka má limitované zdroje a možnosti, lze dosáhnout lepších výsledků při využití agregace většího počtu jednotek [4].

## 3.1 IEEE802.15.4

Jedná se o standard pro LR-WPAN (Low-rate wireless personal area networks), jednoduchou a levnou komunikační síť, která umožňuje komunikaci aplikací s omezeným napájením a sníženými nároky na propustnost [5]. Standard se snaží dodržet architekturu ISO/OSI (International Standards Organization/Open Systems Interconnection) a definuje pouze první dvě vrstvy, jmenovitě fyzickou vrstvu a vrstvu přístupu k médiu. Díky tomu slouží jako základ pro další protokoly, které jej rozšiřují a definují jednotlivé vyšší vrstvy.

Fyzická vrstva (PHY) má na starost přístup k fyzickému HW, jeho aktivaci a deaktivaci. Důležitou částí je výběr vhodného kanálu, který je nezbytný, protože standard definuje několik kanálů pro různá pásma. V neposlední řadě také příjem a odesílaní samotných dat. S tímto souvisí energetická náročnost, která je řešena právě na fyzické vrstvě [5].

Vrstva přístupu k médiu (MAC) umožňuje využití speciálních beacon rámců. Kontrola přijatých dat a jejich potvrzení, přístup k médiu a metody, které umožňují komunikaci s vrstvou za účelem přidání zabezpečovacích mechanizmů ve vyšších vrstvách [5].

Standard zahrnuje definice dvou typů zařízení, první FFD (Full-function device), které může v síti operovat jako koordinátor. Druhý typ, RFD (Reduced-function device), naopak může působit pouze jako bod v síti a je určen především pro velice jednoduché aplikace, které jsou většinou propojeny pouze s jedním FFD.

Podporované topologie jsou hvězda nebo spojení bod-bod. V závislosti na povaze aplikace lze využít kteroukoliv z nich, nebo jejich kombinaci. V zapojení do hvězdy musí síť obsahovat alespoň jednoho koordinátora, který slouží jako primární zdroj kontaktu a je většinou připojen na napájení z elektrické rozvodné sítě. Spojení bod-bod také obsahuje koordinátora, avšak jeho úkol je odlišný, protože v této síti je každé zařízení schopno komunikovat s kterýmkoliv jiným, pokud je v dosahu [5].

## 3.2 ZigBee

Jedná se o technologii, která byla vytvořena jako rozšíření pro IEEE802.15.4. Pro své operace v Evropě ZigBee využívá pásmo 868 MHz a 2,4 GHz. Síť postavená na této technologii je navržena tak, aby byla schopna do určité míry opravit výpadky spojení. Body jsou spojeny do mesh sítě, tzn. každý bod je redundantně spojen s více dalšími body. Projekt je zastřešen společností Zigbee Alliance, založenou v roce 2002 [6].

Jak již bylo řečeno, ZigBee je postaveno na IEEE802.15.4, dodržuje tedy definici spodních vrstev a přidává dvě své vrstvy, které jsou navzájem oddělené za použití SAP (Service Access Point). Vrstva nad MAC se nazývá síťovou vrstvou (NWK). Má na starost především řízení mesh sítě, to zahrnuje všesměrové vysílání sítí, rozhodování o výběru cesty pro data, která putují jednosměrně, a obecné rozhodování o tom, přes jaké body budou data putovat do cíle. Dále vrstva obsahuje příkazy pro zabezpečení spojení. Komunikace přes ZigBee je zabezpečena již na síťové vrstvě, data putují šifrovaná do nižších vrstev. Šifrování je provedeno pomocí 128 bitového AES (Advanced Encryption Standard) [7].

Nejvyšší aplikační vrstva je rozdělena na dvě části. První část (APS) slouží jako komunikační bod pro konečné aplikace, především pro zjednodušení logiky. Před odesláním dat do nižších vrstev kontroluje, zda-li je aplikace součástí nezbytných profilových skupin. Filtruje duplicitní data, která mohla být obdržena ze síťové vrstvy, a udržuje tabulku spojení, ve které jsou uvedeny body nebo skupiny se kterými daná aplikace komunikuje. Druhá část se nazývá ZDO (ZigBee Device Object). Je zodpovědná za stav bodu v síti, objevování dalších spojení nebo skupin v síti [7].

ZigBee se zaměřuje především na co nejnižší spotřebu energie. Toho je dosaženo pomocí velmi krátkých vysílacích intervalů, protože síť nepotřebuje konstantně vysílat data k udržení spojení. Sítě s touto technologií bývají většinou velmi tiché, to zabraňuje i nadbytečnému rušení přenosového pásma [7].

Zařízení jsou dostupná jako moduly, které lze připojit k mikrokontroléru za pomoci známých datových sběrnic. Dále jako SoC (System on Chip), které kombinují mikrokontrolér přímo propojený s rádiovou částí ZigBee [6].

## 3.3 6LowPAN

Jedná se o sadu standardů definovanou IETF (Internet Engineering Task Force) skupinou, která se také stará o všechny klíčové Internetové standardy. Pohled na standard, jeho cíle a předpoklady jsou uvedeny v RFC 4919 [8]. Podobně jako ZigBee je i tento protokol postaven nad IEEE 802.15.4.

Základním prvkem 6LowPAN sítě je tzv. stub síť, která může přijímat a vysílat datové rámce, avšak nemůže sloužit k přenosu dat do jiných sítí. Stub lze rozdělit na tři hlavní typy [9]:

- Ad-hoc - Není připojena k Internetu a umožňuje komunikaci zařízení pouze mezi sebou.
- Jednoduchá (Simple) - Je připojena k IP (Internet Protocol) síti pomocí jednoho okrajového směrovače (LoWPAN Edge Router).
- Rozšířená (Extended) - Využívá více okrajových směrovačů, které jsou navzájem propojené většinou pomocí IP sítě.

Jelikož 6LowPAN vychází z IP protokolu má velmi podobnou strukturu, viz obrázek 3.1. V praxi to znamená, že zařízení implementuje LoWPAN spolu s IPv6 protokolem, proto oba lze vnímat jako součást síťové vrstvy. Nejpoužívanější protokol transportní vrstvy je UDP (User Datagram Packet), především z důvodu jednoduchosti implementace a nenáročnosti na výpočetní výkon. Převod mezi úplným IPv6 protokolem a LoWPAN zajišťují okrajové směrovače [9].

Adresování je naprosto stejné jako v případě IPv6, kdy je adresa formovaná automaticky na základě prefixu sítě a adresy linkové vrstvy (MAC). Rozdíl u LoWPAN spočívá v přímém mapování mezi linkovou adresou a IPv6, tím lze dosáhnout určité komprese. V praxi to znamená, že pro adresování je využito 8 - 16 bitů. K ušetření prostoru standard využívá také komprese hlaviček, kdy při plné kompresi lze kombinaci IPv6 a UDP snížit ze 48 na 6 bytů. Ve výsledku je možno zvýšit počet dat, které lze přenést najednou [9].

Standard také definuje možnost automatické konfigurace sítě pomocí 6LoWPAN-ND (6LoWPAN Neighbor Discovery), které vychází ze stejnojmenné technologie pro IPv6. Protokol je využit po připojení nového zařízení do sítě v okamžiku, kdy je možná jednosměrná komunikace s jakýmkoliv zařízením v síti. Takto zformovaná síť se bude skládat z hostů (Host) a směrovačů (Router). Pouze směrovač může přeposílat data jiným zařízením. Pokud v síti existuje okrajový směrovač, stará se o mnohem složitější funkce, jako je překlad paketů na IPv4, a v případě potřeby jejich směrování mimo LoWPAN síť a uchovává tabulku zařízení, která jsou dostupné v síti [9].

## 3.4 Wi-Fi

Jedná se o velmi rozšířenou bezdrátovou technologii, která je navržena pro přímé připojení zařízení k IP sítím. Celá specifikace je zahrnuta v označení IEEE 802.11 [10] a obsahuje několik standardů, které upravují nebo přidávají nové možnosti komunikace. Písmenná označení jako dodatky byla sjednocena pod Wi-Fi X, kde X udává pořadové číslo, z důvodu zmatení široké veřejnosti.


Obr. 3.1: Porovnání protokolu TCP/IP a 6LoWPAN [9]

Wi-Fi využívá pro přenos pásma 2,4, 5 a 60 GHz. Pásmo 2,4 je rozděleno na 13 kanálů s odstupem 5 MHz a šířkou 22 MHz. Toto je problematické, protože tímto způsobem vznikají pouze tři vzájemně se nepřekrývající kanály a dochází ke značnému rušení. Situace v pásmu 5 GHz je lepší, protože žádné z 25 kanálů se nepřekrývají (za předpokladu použití 20 MHz šířky) [11].

Zaměření této technologie je především na rychlost přenosu dat a nikoliv na bateriově napájené aplikace. Wi-Fi disponuje módem pro šetření energie. Jakékoliv zařízení může operovat ve dvou režimech, a to aktivní, kdy je udržováno stálé spojení a poslech linky, a úsporný, kdy zařízení kontaktuje své AP (Access Point), které udržuje tabulku s informacemi, kdo je v úsporném módu. AP poté ukládá všechny zprávy pro dané zařízení, která se periodicky probouzí [11].

## 3.5 Bluetooth

Podobně jako Wi-Fi je Bluetooth velmi rozšířené a je určeno pro sítě malých rozsahů. Bylo vyvinuto především pro komunikaci mezi dvěma zařízeními s menšími energetickými nároky a pro přenos dat menší rychlostí. Celá specifikace se zastřešena standardem IEEE802.15.1 [12]. Bluetooth protokol je velmi atypický v porovnání s jinými IEEE protokoly. Definuje mnoho dalších vrstev nad fyzickou (PHY) a linkovou (MAC), avšak některé z nich jsou nepovinné. To umožňuje rozdělení do velmi specifických profilů, které jsou zaměřeny na konkrétní aplikace [11].

Architektura sítě je Master-Slave, kdy Master řídí celou komunikaci, je zodpovědný za alokaci kanálu pro dané Slave zařízení a průběžné dotazování, zda-li Slave potřebuje komunikovat. Z toho plyne mírná nevýhoda, Slave zařízení spolu nemohou přímo komunikovat v rámci jedné sítě. Takto vytvořené celky se v rámci Bluetooth nazývají piconet [11].

Bluetooth komunikuje v pásmu 2,4 GHzs šířkou 1 MHz, k dispozici je tedy 79 nepřekrývajících se kanálů. K předcházení rušení se používá metoda FHSS (Frequency Hopping Spread Spectrum), neboli přeskakování mezi jednotlivými kanály. V případě Bluetooth se jedná o 1600 skoků za sekundu. Je definováno několik tříd vysílacích výkonů s určitým dosahem, které jsou uvedeny v tabulce 3.1 [13].

Tab. 3.1: Přehled tříd Bluetooth zařízení [13]

| Třída | Maximální povolený výkon | Přibližný dosah |
|---|---|---|
| - | mW | m |
| Class 1 | 100 | 100 |
| Class 2 | 2,5 | 10 |
| Class 3 | 1 | 1 |

Za účelem snížení spotřeby vznikl BLE (Bluetooth Low Energy), který využívá především mnohem kratší aktivní vysílací doby. To umožňuje výrazné snížení spotřeby oproti starším verzím. Lze zde také využít všesměrového vysílání pro případy, kdy není potřeba, aby vysílač dostal zpět data, nebo pokud je více zařízení příjemcem stejných dat. Menší nevýhodou je, že uživatelská data jsou při broadcastu omezena na velikost 31B.

## 3.6 RFM69

Nejedná se o technologii ani standard, ale SoC, které v sobě integruje vysílač i přijímač. K dostání jsou různé moduly s tímto SoC, obsahující všechny nezbytné komponenty pro samostatné fungování. Systém sám o sobě nepodporuje žádný rozšířený standard. Komunikace probíhá skrze SPI (Serial Peripheral Interface) sběrnici zápisem do určených registrů. Obsahuje podporu 433 a 868 MHz frekvenčního pásma. Velkou výhodou je velmi malá spotřeba energie, během režimu spánku dosahuje maximálně 100 nA. Základní modul může dosáhnout až 13 dBm výstupního vysílacího výkonu s možností připojení externí antény [14].

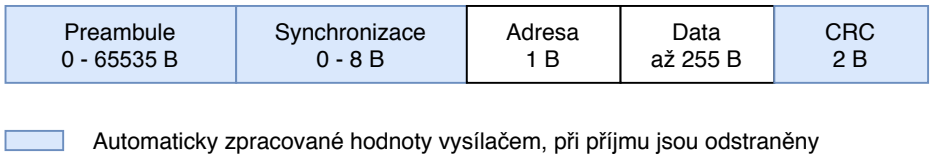
### Komunikace

Komunikace po SPI sběrnici probíhá v módu 0. To znamená, že hodnota klidového hodinového signálu je v úrovni 0, a bit je čten při přechodu vzestupné hraně hodinového signálu. Rozlišení mezi zápisem a čtením registru je indikováno pomocí nejvyššího bitu adresy - 1 určuje, že se jedná o zápis, a 0 o čtení. Zařízení podporuje tři módy komunikace [14]:

- Jednotlivě (Single) - Adresový byte, který následuje byte dat, která mají být zapsána, nebo data vyčtená z daného registru.
- Dávkově (Burst) - Adresový byte, nasledovaný několika byty pro čtení nebo zápis, a adresa registru je automaticky inkrementována vnitřní logikou obvodu.
- Přístup do FIFO (FIFO) - Pokud je adresovaný registr FIFO (First in First out), adresa je inkrementována pouze vnitřně a dochází ke čtení nebo zápisu pouze v rámci FIFO.

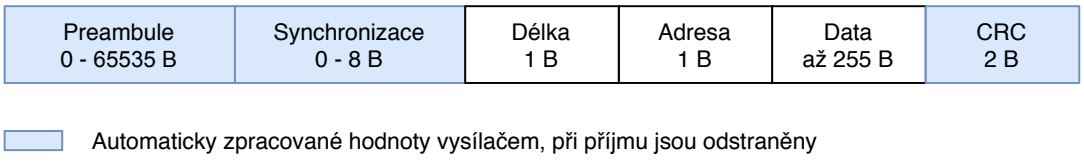
### Formát paketů

RFM69 podporuje tři formáty paketů. Fixní formát je vhodný k minimalizaci určité režie dat. Není třeba posílat délku zprávy ani ji vypočítávat, viz obrázek 3.2. Přenos funguje pouze za předpokladu, že jsou obě komunikující strany nastaveny na stejnou hodnotu [14].




Obr. 3.2: RFM69 fixní formát paketu [14]

Proměnná délka umožňuje posílat zprávy různé délky, kdy první byte paketu obsahuje jeho celkovou délku, viz obrázek 3.3. Oba formáty podporují maximální délku zprávy až 255 bytů za předpokladu, že se jedná o nezabezpečený přenos. Pro zabezpečený je maximální délka limitována na 64 bytů. Zabezpečení přenosu je řešeno přímo v modulu s využitím 128 bitového AES, kdy jsou šifrována uživatelská data [14].




Obr. 3.3: RFM69 proměnný formát paketu [14]

Poslední přenos je s neomezenou délkou, kdy vysílání probíhá kontinuálně a je na uživateli, aby kontroloval detekci úplných dat s případnými detekcemi chyb [14].

Pro první dva zmíněné módy je podporována kontrola přijatých dat pomocí 16 bitového CRC (Cyclic Redundancy Check). V závislosti na nastavení je možnost poškozená data přijmout nebo zahodit při první kontrole [14].

### Podpora přerušení

Modul disponuje 6 výstupními piny, které je možno namapovat na různá přerušení dle potřeby, případně výstup z hodinového signálu modulového oscilátoru. Velmi důležitá je indikace nově přijatých dat. Není tedy nutné nepřetržitě sledovat registry, zda-li jsou k dispozici nová data [14].

## 3.7 Porovnání

Z výše uvedených je k přenosu vybráno RFM69. Je to především kvůli velmi nízké spotřebě ve spánku a velké možnosti konfigurace. Další výhodou je cena, která činí přibližně 100 Kč za kus. Poslední nesporná výhoda je práce v pásmu pod 1 GHz, a to především z důvodu rušení. Pásma, na kterých lze s RFM69 vysílat, nejsou tak zarušená jako 2,4 GHz. Nevýhodou může být, že nemá žádný standardizovaný protokol, který by umožňoval jednoduché napojení na již existující síť.

# 4 Použité technologie

## 4.1 ARM

Jedná se o architekturu procesorů, které jsou využívány v mnoha aplikacích především pro svoji nízkou spotřebu energie. ARM architektura využívá techniky k podpoře velkého množství implementací ARM procesorů. Definuje základní instrukční sadu, modely vyjímek (Exception) spolu s modelem paměti. Mikroarchitektura určuje, jakým způsobem implementace dodrží tento model [15].

Tab. 4.1: Přehled architektury ARM [18]

| Architektura | Adresová/datová sběrnice | Jádro | Poznámka |
|---|---|---|---|
| ARMv1 | 26/32 bitů | ARM1 | Technologické demo |
| ARMv2 | 26/32 bitů | ARM2, ARM3 | HW násobička a MMU |
| ARMv3 | 26/32 bitů | ARM6, ARM7 | |
| ARMv4 | 26/32 bitů | ARM8 | |
| ARMv5 | 26/32 bitů | ARM7EJ, ARM9E, ARM10E | Starší zařízení |
| ARMv6 | 32/32 bitů | ARM11 | Dodnes používané, např. Raspberry PI Zero |
| ARMv6-M | 32/32 bitů | Cortex-M0, Cortex-M0+, Cortex-M1 | Mikrokontroléry |
| ARMv7-M | 32/32 bitů | Cortex-M3 | Mikrokontroléry |
| ARMv7E-M | 32/32 bitů | Cortex-M4, Cortex-M7 | Mikrokontroléry |
| ARMv7-R | 32/32 bitů | Cortex-R4, Cortex-R5, Cortex-R7 | Realtime aplikace |
| ARMv7-A | 32/32 bitů | Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9, Cortex-A12, Cortex-A15, Cortex-A17 | Výkonné procesory, např. Raspberry PI 2 |
| ARMv8-A | 32/64 bitů | Cortex-A53, Cortex-A57, Cortex-A72 | Výkonné procesory, např. Raspberry PI 3 a 4 |

### 4.1.1 Rodiny procesorů

Postupem času ARM vyvinul velké množství procesorů, z tohoto důvodu bylo třeba lépe odlišit jednotlivé rodiny. Proto vznikla značka Cortex, která je rozdělena na tři hlavní skupiny [16].

#### Cortex-A

Jedná se o aplikační procesory se zaměřením na vysoký výkon a podporu různých operačních systémů. Většina těchto procesorů má 32 bitovou datovou linku, v posledních letech se vyrábí i procesory s podporou 64 bitů. Nabízejí také virtuální paměť za použití MMU (Memory Managment Unit), rozšířenou podporu programovacího jazyka Java a zabezpečené prostředí pro běh programů tzv. TrustZone. Tato řada procesorů pracuje na relativně vysoké taktovací frekvenci, většinou v řádech jednotek GHz [16].

#### Cortex-R

Řada výkonných procesorů, které jsou zaměřeny na zpracování velkého množství dat v reálném čase. Podobně jako Cortex-A řada běží na vysokých taktovacích frekvencích v rozsahu 500 MHz až 1 GHz. Obsahují cache paměť, která umožňuje předvídatelné chování obsluhy přerušení. Podporují také dodatečné prvky, které zvyšují spolehlivost těchto procesorů, například ECC (Error Correction Code) pro paměť. Dále logiku pro lepší odhalování chyb běhu programu. Toto vybavení dělá z těchto procesorů vhodné kandidáty pro diskové nebo modemové řadiče, a dále také specializované obvody pro průmyslové využití [16].

#### Cortex-M

Poslední řada je určena pro aplikace, kde není kritický výpočetní výkon, ale jsou kladeny vysoké nároky na spotřebu energie. Jako výsledek těchto optimalizací je výrazně snížená taktovací frekvence, která ve většině případů nepřesahuje 100 MHz. Procesory jsou schopny velmi rychle a předvídatelně odpovědět na přerušení. Toho je dosaženo pomocí velmi úzkého spojení mezi výpočetní částí a NVIC (Nested Vectored Interrupt Controller). Procesory z této řady jsou kombinovány do SoC jako kontroléry pro ovládání jiných částí např. bezdrátové komunikace [16].

### 4.1.2 Architektura

#### Registry

ARM disponuje 31 registry, které jsou určeny pro běžné použití. V kterýkoliv okamžik je viditelných 16 registrů, zbytek je použit ke zrychlení výpočetního procesu. Tři z viditelných mají speciální roli [17]:

- SP (Stack pointer) - Software běžně používá R13 jako SP.
- LR (Link register) - R14 obsahuje adresu další instrukce po použití větvení programu, dále návratovou hodnotu po vstupu do Exception módu.
- PC (Program counter) - R15 je běžně používán jako ukazatel na instrukci, která se nachází přesně dvě instrukce po té, co je právě vykonávána.

#### Exceptions

Ve chvíli, kdy se objeví Exception, ARM přeruší probíhající operaci definovaným způsobem a zahájí výpočet na fixní předem stanovené adrese. Tyto adresy se nazývají Exception vektory (Exception vectors). Za běžných podmínek programy vloží instrukce, které se mají vykonat, na tato předem stanovené místa. Je podporováno sedm druhů Exception [17]:

- Reset
- Pokus o vykonání neznámé instrukce
- SWI (Software interrupt), softwarové přerušení
- Prefetch abort
- Data abort
- IRQ (Interrupt request)
- FIQ (Fast interrupt)

#### Intrukční sada

Instrukční sada je rozdělena na šest širších skupin [17]:

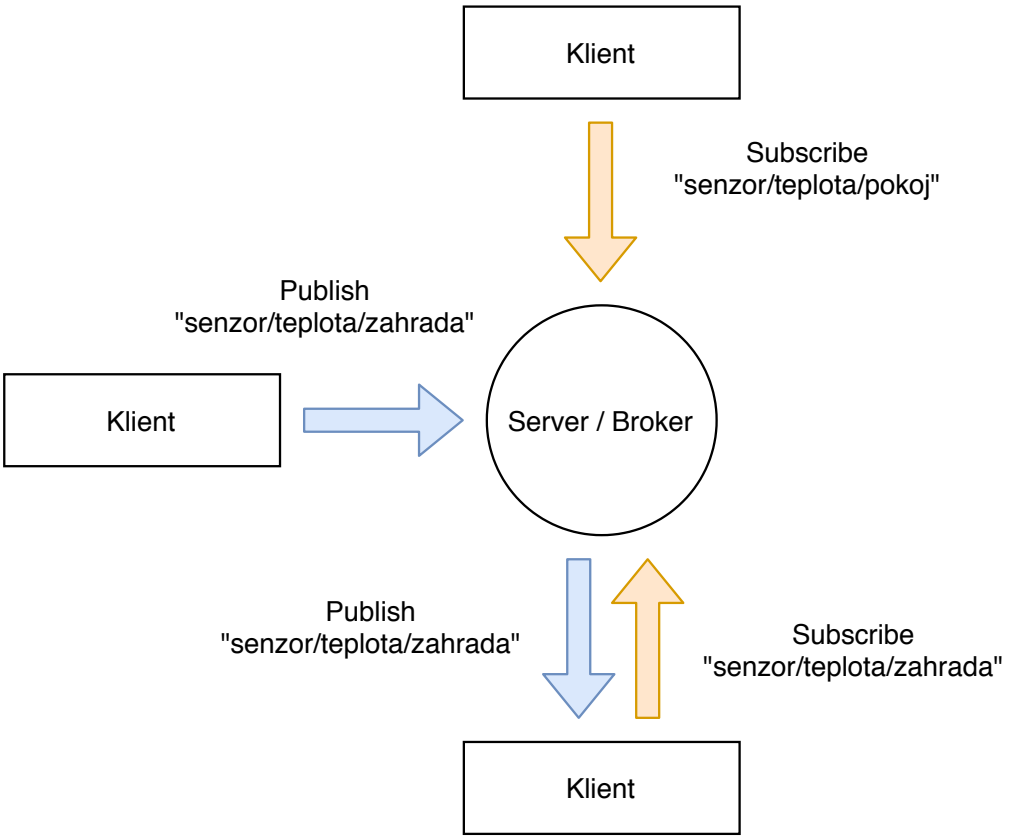
- Instrukce pro větvení programu (Branch instructions)
- Instrukce pro zpracování dat
- Instrukce pro operace se stavovým registrem (Status register)
- Instrukce pro ukládání (Store) a načítání (Load)
- Instrukce pro koprocesor
- Instrukce pro generování Exceptions

## 4.2 MQTT

MQTT (Message Queuing Telemetry Transport) je protokol na komunikaci pomocí publish/subscribe modelu, viz obrázek 4.1. Je navržen tak, aby byl velmi jednoduchý a rychlý. Je vhodný pro použití u všech zařízení, která mají omezené možnosti připojení a vysoké nároky na spotřebu energie [19]. Celý standard je zastřešen společností OASIS. K tomu aby mohl fungovat model publish/subscribe je zapotřebí klientské aplikace (Client) a serverové části (Broker).

### Komunikace

Komunikace probíhá skrze Broker, každý klient musí být připojen. Klienti se mohou přihlásit k topicu, který je zajímá, nebo poslat zprávu na daný topic. Obě věci lze dělat naprosto nezávisle na sobě. To znamená, že může existovat implementace klienta, která pouze odesílá zprávy a nepřijímá je, nebo naopak. Každá zpráva, která dorazí na server, je filtrována a poté rozeslána pouze klientům, kteří mají zájem o daný topic [20].




Obr. 4.1: Model MQTT publish/subscribe [20]

### Topic

K odeslání zprávy musí být vždy specifikován topic, který má ve velkém množství případů následující formát *senzor/teplota/zahrada*. Tuto strukturu lze přirovnat k souborovému systému, kde je možno topic formovat do různých úrovní a rozdělit jej tak dle relevance, popřípadě, pokud neznáme všechny topicy v dané úrovni, lze využít zástupný znak. MQTT podporuje dva zástupné znaky pro topic. Výhoda je, že lze odeslat pouze jednu subscribe zprávu a obdržet relevantní data [20].

Zástupný znak *+* umožňuje nahradit kteroukoliv úroveň topicu a znamená, že klient má zájem o vše co by mohlo nahradit *+*. Například *senzor/+/zahrada* říká, že pokud máme tři senzory v úrovni *zahrada*, chceme obdržet zprávy ze všech: *senzor/teplota/zahrada*, *senzor/vlhkost/zahrada*, *senzor/tlak/zahrada* [20].

Druhý zástupný znak *#* lze použít pouze na konečné úrovni, tedy *senzor/teplota/#*. Tento znak umožňuje, podobně jako *+*, obdržet všechny zprávy na úrovni kterou zastupuje, ale také úrovně následující po nahrazené. Například tedy *senzor/teplota/zahrada*, *senzor/teplota/zahrada/F*, *senzor/teplota/pokoj*, *senzor/teplota/pokoj/F*. Z toho plyne že použití *#* na první úrovni je zástupný znak pro všechny topicy, kterými daný broker disponuje [20].

### QoS

QoS (Quality of Service), neboli kvalita služeb, je důležitý aspekt v MQTT. Udává, jakým způsobem jsou doručována data a jejich potvrzení. Rozeznáváme tři třídy QoS [20]:

- 0, Nejvíce jednou (At most once) - Tato úroveň nezaručuje nic navíc proti TCP (Transmission Control Protocol), který se používá pro přenos dat. Data jsou odeslána klientskou stranou na server, nic jiného se neděje. Velkou výhodou je minimální režie.
- 1, Nejméně jednou (At least once) - Oproti předchozí úrovni se přidává základní potvrzení doručení. Tímto způsobem je zaručeno že zpráva bude doručena nejméně jednou. Nevýhodou je, že takto lze vytvářet duplicitní data. Pokud nedojde potvrzení od příjemce v určitém časovém limitu jsou data znovu poslána.
- 2, Přesně jednou (Exactly once) - Poslední úroveň eliminuje problém s duplikováním dat. Je zde navíc další výměna potvrzení. Nevýhodou je poměrně velké množství režijních dat, které je třeba poslat.

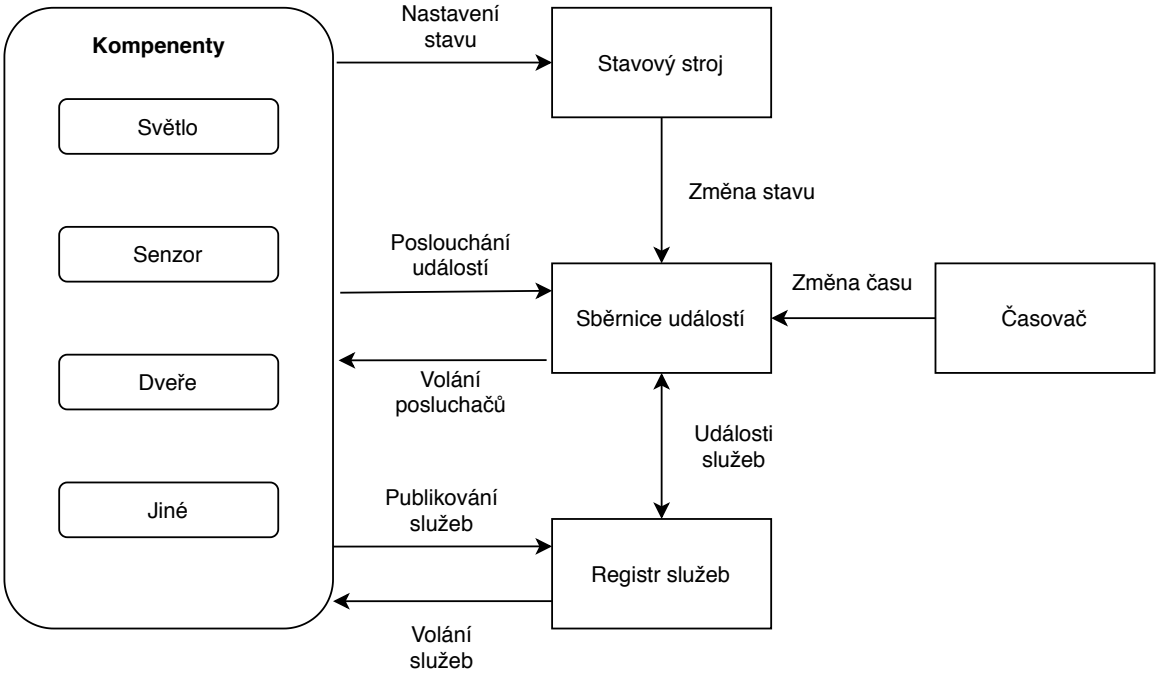
### Zabezpečení

Většina MQTT služeb komunikuje ve výchozím nastavení po nezabezpečeném TCP portu 1883. To znamená, že kdokoliv po cestě paketu může vidět jejich obsah, stejně

jako všichni s přístupem na server se mohou přihlásit k libovolnému topicu nebo odesílat data. První způsob zabezpečení je šifrování samotných dat. Standard má rezervovaný TCP port 8883, pomocí kterého lze použít TLS (Transport Layer Security). Toho lze dosáhnout vytvořením certifikátu pro server a vynucením jeho použití. Tento způsob zajistí, že nikdo po cestě nemůže přečíst data. Druhý způsob je autorizace klientů pomocí certifikátu nebo hesla. Tímto lze předejít druhému problému, pouze klient se správným certifikátem/heslem je schopný se připojit na server. Důležitá je kombinace obou způsobů pro zajištění dobré úrovně zabezpečení [20].

## 4.3 Home Assistant

Jedná se o Open Source projekt pro domácí automatizaci, který klade kontrolu a soukromí. Zahrnuje jednoduché webové rozhraní, pomocí kterého lze vše monitorovat a konfigurovat. Demo aplikace je dostupné na adrese `https://demo.home-assistant.io`. Home Assistant podporuje velké množství zařízení skrze rozhraní zvané Integrace (Integration) [21].




Obr. 4.2: Vnitřní architektura Home Assistant [22]

### 4.3.1 Architektura

Architektura je rozdělena na čtyři hlavní části, které dohromady tvoří jádro projektu, viz obrázek 4.2 [22]:

- Sběrnice událostí (Event Bus) - Zodpovědná za všechny události, které přijdou nebo jsou třeba odeslat, jedná se klíčovou část celého systému.
- Stavový stroj (State machine) - Udržuje stav všech komponentů a posílá událost o změně stavu v případě potřeby.
- Registr služeb (Service registry) - Poslouchá události, které si žádají nebo chtějí registrovat novou službu.
- Časovač (Timer) - Každou vteřinu posílá událost o změně času.

#### Komponenty

Komponenty (Components) jsou hlavní částí pro integraci HW zařízení. Jedná se o idealizovaný model. Home Assistant v základu obsahuje několik základních komponentů, na kterých lze stavět ostatní [22]:

- Kvalita vzduchu (Air Quality)
- Kontrolní panel alarmu (Alarm Control Panel)
- Binární senzor (Binary Sensor)
- Ovládání klimatu (Climate)
- Krytí (Cover)
- Ventilace (Fan)
- Světlo (Light)
- Zámek (Lock)
- Multimediální přehrávač (Media Player)
- Senzor (Sensor)
- Přepínač (Switch)
- Vysavač (Vacuum)
- Ohřívač vody (Water Heater)
- Počasí (Weather)

## 4.4 Rust

Jedná se o programovací jazyk, který se snaží umocnit vývoj dostupnými nástroji. Hlavní zaměření jazyka je bezpečný kód při napsání. Toho se snaží docílit eliminací věcí, které působily problémy v jazycích jako například C/C++ [23]. Vývoj započal v roce 2010 pod hlavičkou Mozilla Research. Syntakticky se kód velice podobá jazykům vycházejícím z C.

### 4.4.1 Ownership

Nejvíce unikátní vlastnost jazyka je Ownership. Všechny programy se musí určitým způsobem starat o paměť, se kterou pracují. Některé jazyky mají GC (Garbage Collector), který konstantně sleduje využití paměti a uvolňuje nevyužívané místo. Jiné jazyky se musí o paměť starat manuálně. Pokud je prostor potřeba musí jej alokovat a následně zase uvolnit. Rust využívá jiného přístupu, a to Ownership pravidel, která jsou kontrolována při kompilaci. Žádné z těchto pravidel nezpomaluje výsledný program. Pravidla jsou celkem tři a jejich koncept je jednoduchý [23]:

- Každá hodnota je přiřazena do proměnné, která je majitel (owner) této hodnoty.
- V daném okamžiku může existovat pouze jeden majitel.
- Pokud se majitel dostane z dosahu programu (Out of Scope) jeho hodnota bude zahozena, tím pádem uvolněna paměť, kterou zabíral.

Způsobů, jakými si lze předávat data v programovacím jazyku Rust, je několik.

#### Move

První důležitý je přesun (Move). Přesunutím hodnoty do nové proměnné (owner), tímto není původní proměnná platná a jakýkoliv přístup k ní je striktně odmítnut [23].

Výpis 4.1: Rust Move

```
fn main() {
  let x = 5;  // Přiřazení hodnoty 5 do proměnné x
  let y = x;  // Přesunutí hodnoty 5 do proměnné y, proměnná x
 od tohoto bodu není platná
}
```

#### Copy

Druhá možnost je kopírování (Copy). Jak již název napovídá, hodnota v původní se zkopíruje do nové proměnné, tím dojde k alokaci nového místa v paměti. Obě proměnné jsou platné a mohou nezávisle nakládat se svojí hodnotou [23].

Výpis 4.2: Rust Copy

```
fn main() {
  let x = 5;  // Přiřazení hodnoty 5 do proměnné x
  let y = x.clone();  // Kopírování hodnoty 5 do proměnné y,
 proměnná x je stále platná
}
```

### Borrow

Poslední možnost je vypůjčení (Borrow). Aby byla dodržena pravidla přístupu k paměti a nemohlo docházet k situacím, kdy dvě části programu mohou zapisovat do stejné proměnné, v danou chvíli může být pro zápis pouze jedno vypůjčení. Pokud není proměnná vypůjčená pro čtení, lze je půjčit nekonečně mnohokrát [23].

Výpis 4.3: Rust Borrow

```
fn main() {
  let x = 5;  // Přiřazení hodnoty 5 do proměnné x
  let y = &x;  // y má referenci na hodnotu x
  let z~= &x;  // z~má referenci na hodnotu x, ve stejnou dobu
 jako y
}
```

### Mutability

Všechny proměnné jsou v rámci jazyka Rust označeny jako neměnné (immutable). Tyto proměnné lze pouze číst a nelze do nich zapisovat. Pokud chceme proměnnou měnit z programu, je třeba ji explicitně označit jako měnitelnou (mutable) [23].

Výpis 4.4: Rust Mutability

```
fn main() {
  let x = 5;  // Hodnotu v~proměnné x nelze měnit
  let mut y = 10;  // Hodnotu v~y lze změnit
  y = 15;
}
```

# 5 Komunikační protokol

RFM69 nestanovuje žádný protokol vyšší vrstvy. Je tedy nutné definovat jakým způsobem si budou jednotlivé strany vyměňovat data a jejich identifikaci. Vzhledem k povaze komunikace není potřeba složitý protokol, který by umožňoval velmi generickou definici dat. Výsledný návrh je rozdělen na dvě vrstvy, které dohromady splňují požadavky potřebné ke komunikaci. Nejnižší vrstva záleží na nastavení RFM69.

## 5.1 Fyzická vrstva

RFM69 je schopno vysílat v pásmech 433, 868 a 915 MHz. V České Republice není možno použít pásmo 915 MHz, protože je určeno pro aplikace mobilní sítě [24]. Pásma 433 a 868 MHz spadají do *Aplikace nespecifikované SRD*. Lze je využít pro zařízení, která vysílají s krátkým dosahem [25] [26]. Pro komunikaci mezi oběma jednotkami je zvoleno pásmo 433 MHz. Použitá modulace je FSK s kmitočtovým zdvihem 50 KHz. Bitová přenosová rychlost je nastavena na 55 555 bit/s. RFM69 zajišťuje veškerou synchronizaci vysílače a přijímače za použití preambule o délce 3 B, kdy je vysílána sekvence 0xAA (binárně 10101010).

### Legislativa

Je nutno podotknout, že vysílání v tomto volném pásmu je omezeno dle *všeobecné oprávnění č. VO-R/10/12.2017-10 k využívání rádiových kmitočtů a k provozování zařízení krátkého dosahu*. Toto omezení stanovuje, za jakých podmínek lze vysílat. RFM69 v pásmu 433 MHz je zařazeno pod označením *g*. Zařízení v této skupině může vysílat s maximálním výkonem 10 mW ERP (Equivalent Radiated Power) a s klíčovacím poměrem maximálně 10 % (je podíl času, kdy zařízení aktivně vysílá, v rámci jakékoliv jedné hodiny) [27].

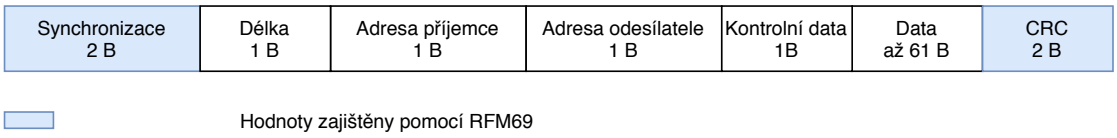
## 5.2 Síťová vrstva

Na této vrstvě je vyřešena adresace jednotlivých zařízení v síti. Formát lze vidět na obrázku 5.1.

### Adresace

Synchronizace je složena ze dvou bytů, první je fixní hodnota 0x2D a druhá je k dispozici jako adresa sítě, v případě práce je nastavena na hodnotu 0x64. Tímto lze dosáhnout filtrace rovnou při příjmu, pokud se synchronizace nebude shodovat, je komunikace ignorována.

Pro adresu zařízení je vyhrazen jeden byte, tímto lze dosáhnout až 256 unikátních adres pro zařízení, které jsou v síti. Paket obsahuje adresu příjemce i odesílatele.




Obr. 5.1: Formát paketu na síťové vrstvě

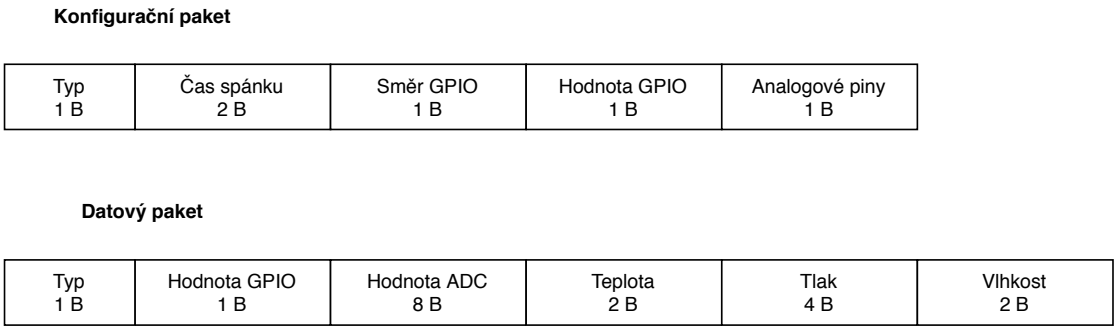
### Kontrolní data

Tato forma paketu umožňuje přenos bez potvrzení i s potvrzením. Za tímto účelem jsou odeslána kontrolní data. V současné podobě lze odeslat pouze tři hodnoty:

- 0x80 - Potvrzení příjmu, ACK (Acknowledgement)
- 0x40 - Žádost o potvrzení příjmu, ReqACK (Acknowledgement Request)
- 0x00 - Přenos bez potvrzení

## 5.3 Aplikační vrstva

Aplikační vrstva nese samotná data aplikace, která jsou rozdělena na dva základní typy. Strukturu lze vidět na obrázku 5.2.




Obr. 5.2: Formát paketu na aplikační vrstvě

### Konfigurační paket

Konfigurační paket má dvě podoby, jedna z nich je žádost o konfiguraci. Tato zpráva obsahuje pouze typ paketu. Do vzdálené jednotky putuje paket již v plném rozsahu, jak jej lze vidět ze struktury.

- Typ - Pro konfigurační paket je hodnota 0x02
- Čas spánku - 16 bitové číslo, které udává spánek v sekundách mezi měřením dat
- Směr GPIO - Udává, jestli budou digitální piny konfigurovány jako vstup nebo výstup
- Hodnota GPIO - Pro výstupní piny určuje jejich hodnotu
- Analogové piny - Které analogové piny se mají měřit

### Datový paket

Paket obsahuje všechna naměřená data a lze jej poslat pouze směrem k bráně.

- Typ - Pro datový paket je hodnota 0x08
- Hodnota GPIO - Hodnota vstupních i výstupních pinů
- Hodnota ADC - 4x 16 bitová čísla s naměřenou hodnotou analogových pinů. Poslední hodnota je naměřené napětí baterie.
- Teplota - Naměřená teplota, 16 bitové číslo
- Tlak - Naměřený tlak, 32 bitové číslo
- Vlhkost - Naměřená vlhkost, 16 bitové číslo

# 6 Vzdálená jednotka

Jak již bylo zmíněno, u vzdálené jednotky je kladen velký důraz na spotřebu. Při návrhu bylo třeba pamatovat na tento fakt, který do určité míry ovlivnil výběr některých komponent a jejich zapojení ve výsledném návrhu na DPS. Výsledný výrobek lze vidět na obrázku 6.1

## 6.1 Výběr komponent

### Napájení

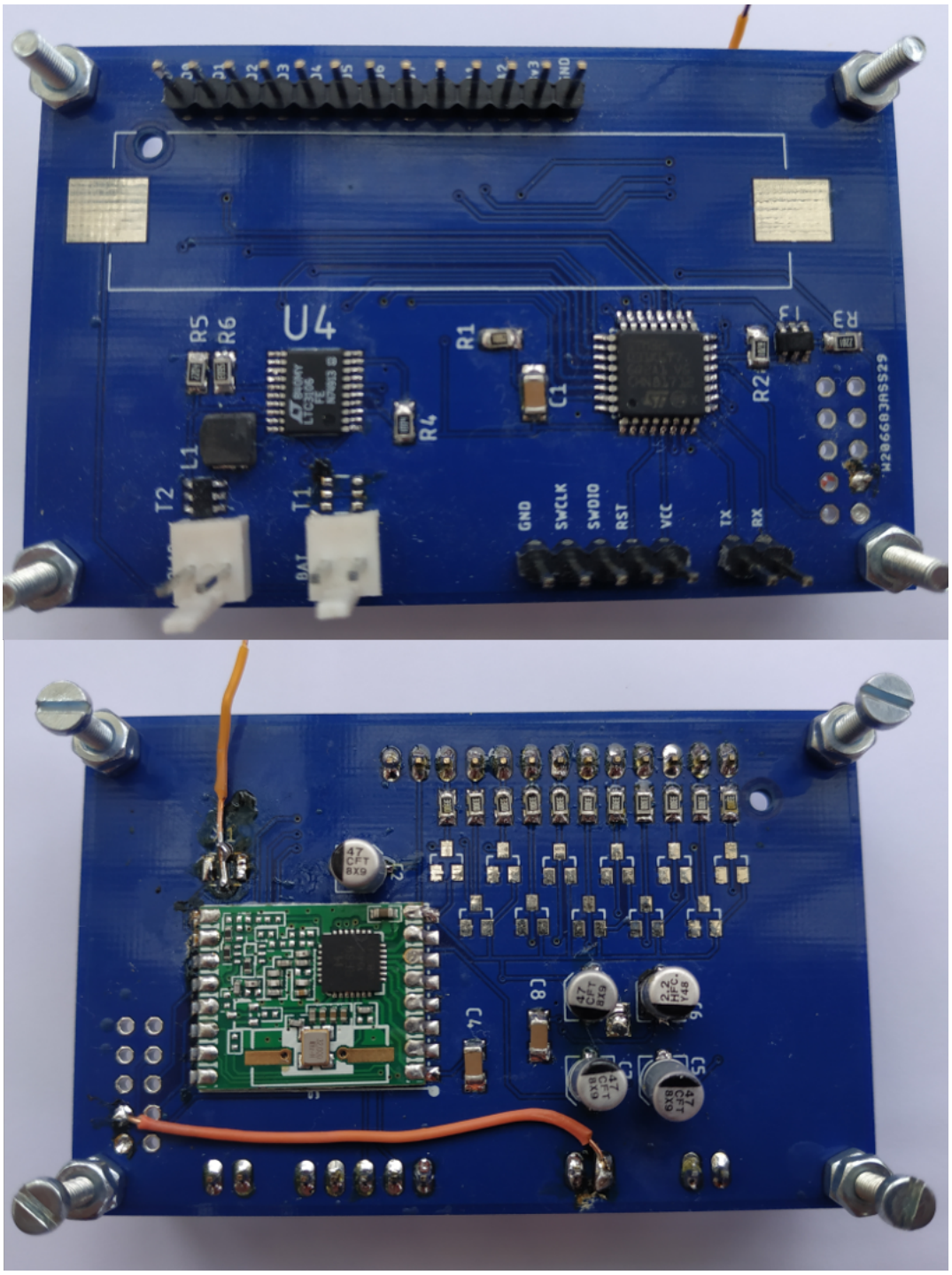
Napájení je řešeno pomocí integrovaného obvodu LTC3106. Obvod kombinuje základní potřebné funkce pro napájení celého řešení. Je schopný nárazově dodat proud až 650 mA, což je více něž dostačující pro napájení vzdálené jednotky. Obvod disponuje dvěma vstupy pro různé zdroje napětí. První vstup je určen pro primární napájecí zařízení, v tomto případě solární článek. Druhý vstup slouží pro přivedení napájení ze záložního zdroje, baterie pro potřeby vzdálené jednotky. K obvodu je nutné dodat jen malé množství pasivních součástek.

Velmi důležitou funkcí je stabilizace výstupního napětí, kterou lze u obvodu konfigurovat. Stabilizace kombinuje dvě metody a to převodu dolů i nahoru. Tato funkce je důležitá především pro napětí 3,3 V, které například solární článek lehce přesáhne při plném venkovním osvětlení. Stejně tak baterie, které jsou obvodem podporované, dosahují hodnot napětí které se pohybuje kolem 3,3 V v obou směrech.

Jak již bylo zmíněno, obvod má dva vstupy a zastává také funkci přepínače vstupů. Pokud je dosaženo na vstupním pinu RUN hodnoty napětí větší jak 600 mV, obvod využívá primární zdroj k napájení (funkce je vybavena hysterezí 50 mV, aby se zabránilo přílišnému přepínání při nestabilitě primárního zdroje). Takto lze docílit napájení celé jednotky ze solárního panelu v případě, že je dostatečně osvětlen. Mimo jiné je při napájení z primárního zdroje je zbytkovým proudem dobíjen zdroj záložní, za předpokladu, že jde o baterii, která umožňuje nabíjení. Tímto lze zvýšit celkovou životnost obvodu na mnohem větší hodnoty.

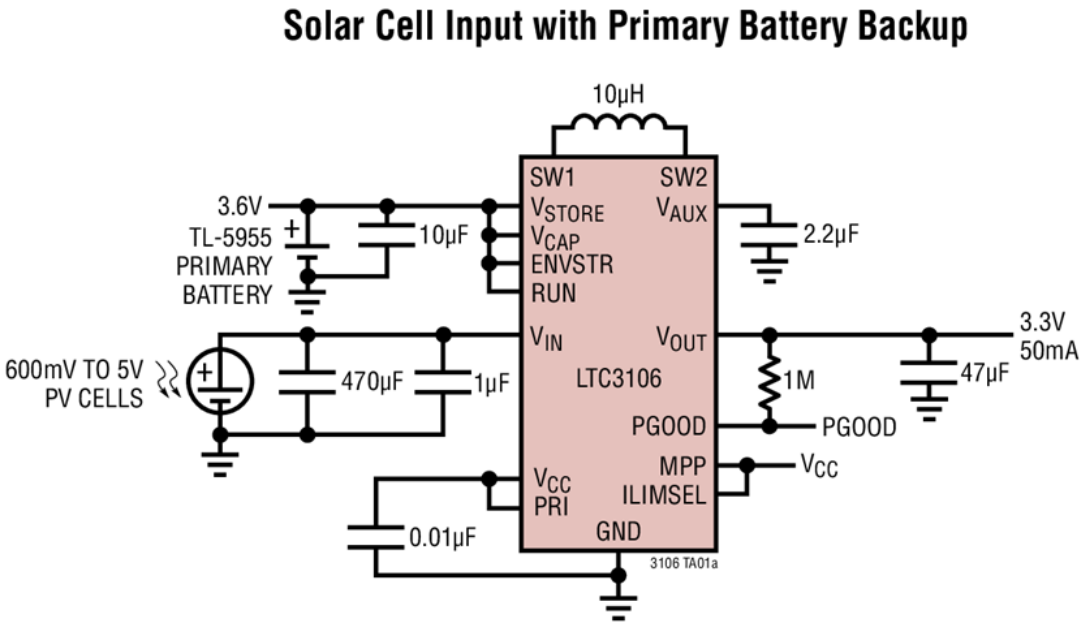
Poslední nespornou výhodou je velmi malý klidový proud 1,6 uA. Obvod by tedy měl zatěžovat baterii naprosto minimálně.

Baterie byla zvolena LiPo (Lithium Polymer). Mezi velké výhody těchto typů baterií patří velká kapacita při malých rozměrech. Nominální napětí jednoho článku je obvykle 3,7 V. Menší nevýhodou je nutná ochrana baterie, především proti nízkému napětí, přebití a vysoké teplotě. Kterýkoliv z těchto negativních vlivů může nenávratně poškodit baterii.




Obr. 6.1: Výsledná vzdálená jednotka

Solární článek je zvolen tak, aby při nominálním napětí byl schopen dodat proud nutný k provozu vzdálené jednotky. Z tohoto důvodu byl zvolen AM-8702CAR od firmy Panasonic, který je schopen dodat proud 34,4 mA při maximálním výkonu.



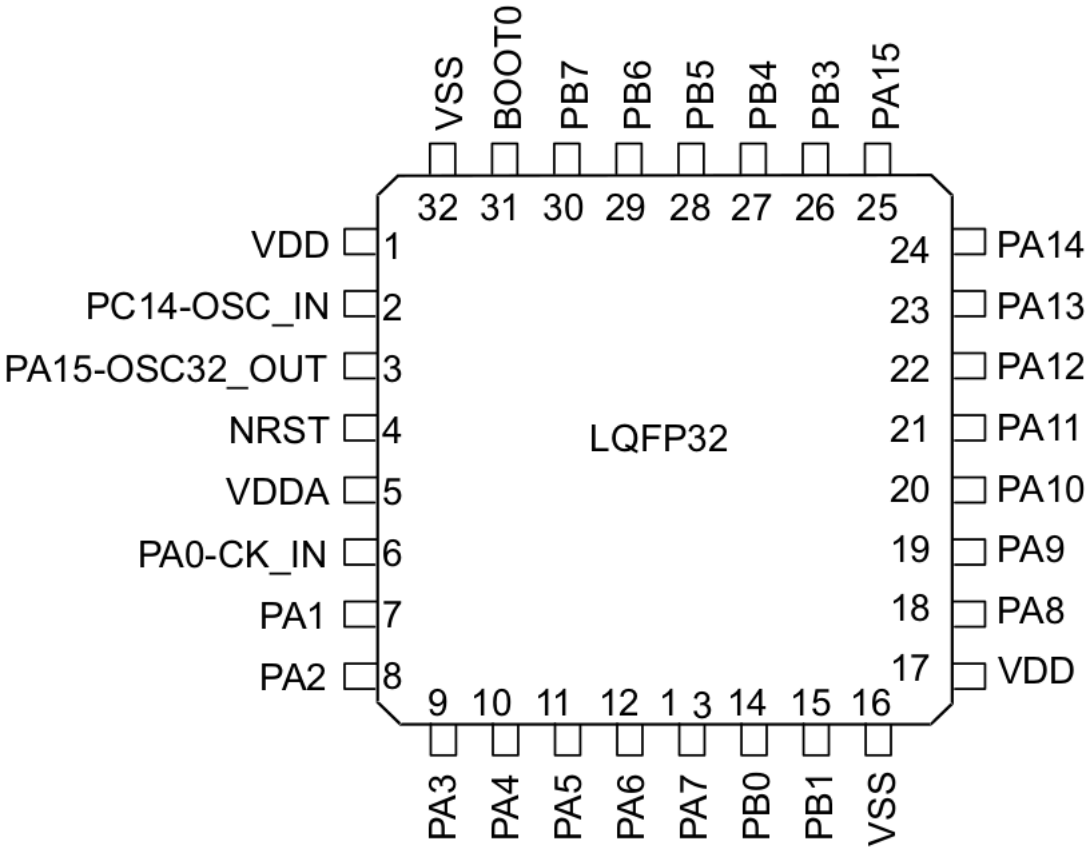

Obr. 6.2: Referenční zapojení obvodu LTC3106 [28]

### Řídící část

K řízení vzdálené jednotky byl použit mikrokontrolér z rodiny STM32, konkrétně STM32L031K6. Jedná se o 32 bitový ARMový procesor s jedním Cortex-M0+ jádrem. Hlavní výhodou je velmi nízká spotřeba ve vztahu k taktovací frekvenci, dle výrobce se jedná o 49 uA/MHz při běhu. Ve spánku pak lze dosáhnout spotřeby přibližně 360 nA při zachování obsahu RAM paměti. Rozložení pinů lze vidět na obrázku 6.3.

STM32L031K6 může pracovat s frekvencí až 32 MHz a obsahuje dva zabudované oscilátory. První oscilátor má frekvenci 16 MHz s přesností 1 %, není tedy nutný žádný jiný externí oscilátor. Frekvenci vnitřního oscilátoru lze zvýšit pomocí PLL (Phase Locked Loop) až na maximální frekvenci. Druhý oscilátor je určen pro operace v režimu nejnižší spotřeby s frekvencí 37 kHz a přesností 10 %. Tento oscilátor lze použít pro RTC (Realtime Clock) během režimu nízké spotřeby.

Mezi důležité periferie, které jsou k dispozici, patří 12 bitové ADC (Analog-Digital Converter) s deseti kanály. Osm kanálů je vyvedeno na externí piny, zbylé dva jsou použity pro vnitřní teplotní senzor a k měření referenčního napětí. Dále je vyvedena jedna SPI a I2C (Inter-Integrated Circuit) sběrnice pro komunikaci s externími zařízeními. Osm časovačů, z toho jeden je již zmíněný RTC, který lze použít pro sledování reálného času, a druhý, s názvem SysTick se běžně používá pro měření času během běhu programu.




Obr. 6.3: STM32L031K6 rozložení pinů [29]

### Měření veličin

K měření základních veličin, tedy teploty, atmosferického tlaku a vlhkosti vzduchu je použito čidlo BME280. Velkou výhodou čidla je integrace v jednom pouzdru, tím pádem není třeba více součástek pro měření. Další výhodou je velice nízká spotřeba 3,6 uA při v běhu a 100 nA ve spánku. S čidlem lze komunikovat pomocí sběrnice I2C nebo SPI. Parametry přesnosti měření jsou uvedeny v tabulce 6.1.

Tab. 6.1: Přehled parametrů BME280

| Typ | Minimální hodnota | Maximální hodnota | Maximální odchylka |
|---|---|---|---|
| Teplota | -40 °C | 85 °C | +-1 °C |
| Tlak | 300 hPa | 1100 hPa | +-1 hPa |
| Vlhkost | 0 % | 100 % | +-3 % |

## 6.2 Schéma zapojení

Schéma zapojení spolu s návrhem DPS pro program Eagle se nachází v adresáři *HW/Node* B. Tato kapitola obsahuje popis jednotlivých částí návrhu. Celkový návrh je ve výsledku jednoduchý, avšak obsahuje pár prvků, jejichž význam je nutné vysvětlit.

### Napájení

Napájecí část vychází z referenčního zapojení LTC3106, viz obrázek 6.2, s několika odlišnostmi. Ve schématu se jedná o část s označením U4. Změna je především u nastavení výstupního napětí, protože celá deska pracuje s 3,3 V. K tomu slouží vstupní piny s označením OS1 a OS2. Zapojení pro různá výstupní napětí je uvedeno v tabulce 6.2. Obvod obsahuje vnitřní referenci s názvem VCC (v tabulce mají taktéž název VCC), ke které se připojují právě tyto typy vstupů, pokud mají být konfigurovány jako logická 1. Ve schématu nesou označení PWR_VCC, protože VCC je určeno pro hlavní napájecí větev celého obvodu.

Tab. 6.2: Konfigurace výstupních napětí LTC3106 [28]

| OS1 | OS2 | Výstupní napětí |
|---|---|---|
| - | - | V |
| GND | GND | 1,8 |
| GND | VCC | 2,2 |
| VCC | GND | 3,3 |
| VCC | VCC | 5 |

Další nastavení je pro typ baterie, která bude k obvodu připojena. Jak již bylo zmíněno, je to nutné z toho důvodu, aby obvod znal hraniční hodnoty napětí a nedošlo k přetížení baterie, a tato aby zároveň mohla být dobíjena správnými hodnotami. Konfigurace nastavení pro různé druhy baterie je uvedena v tabulce 6.3. Pro potřeby vzdálené jednotky je zvolen LiPo článek.

Před vstupem solárního článku je dělič napětí složen z rezistorů R5 a R6. Jak již bylo zmíněno, hlavní zdroj napájení je aktivní v případě napětí většího jak 600 mV na RUN vstupu. Tento dělič má poměr stanoven tak, aby tohoto bylo dosaženo při napětí 3,3 V na výstupu solárního článku. Toto opatření je zavedeno z důvodu, aby byl solární článek schopen dodávat dostatečný výkon k napájení obvodu. Samozřejmě z toho plyne menší dopad na životnost baterie.

Tab. 6.3: Konfigurace baterie LTC3106 [28]

| PRI | SS1 | SS2 | Maximální napětí | Minimální napětí | Typ |
|---|---|---|---|---|---|
| - | - | - | V | V | - |
| GND | GND | GND | 4 | 2,78 | Li Carbon |
| GND | GND | VCC | 2,9 | 1,9 | 2x NiMH |
| GND | VCC | GND | 3 | 2,15 | Li Coin |
| GND | VCC | VCC | 4 | 3 | Li Polymer/Graphite |
| VCC | GND | GND | 4,2 | 2,1 | Primární článek |

Poslední částí napájecího obvodu dělič tvořen rezistory R2 a R3 spolu s tranzistorem T3. Jedná se o část měření napětí baterie řízenou tranzistorem pro co největší šetření životnosti.

### Řídící a rádiová část

Zapojení této části je jednoduché. Jedná se pouze o vyvedení vstupů a výstupů na příslušná místa. Pro komunikace s RFM69 je vyvedena sběrnice SPI. Samotná radiová část obsahuje navíc pouze kondenzátor, který kompenzuje proudové špičky během vysílání dat. Druhá SPI je vyvedena na header spolu s možnosti připojení bateriového vstupu.

Důležitou částí je vyvedení SWD (Serial Wire Debug). Jedná se o sériové rozhraní, které slouží k programování STM32 mikrokontroléru. Rozhraní může sloužit také ke krokování programu přímo za běhu. Jsou k tomu zapotřebí pouze tři vstupy, jmenovitě NRST, SWDIO a SWCLK. Pro potřeby odchytávání zpráv za běhu, bez nutnosti krokování, je vyveden také UART (Universal Asynchronous Receiver Transmitter)

V neposlední řadě se jedná o vyvedení osmi konfigurovatelných digitálních GPIO (General-Purpose Input/Output) pinů na přislušný header. Dva z nich lze použít jako sběrnici I2C, konkrétně D0 a D1, a také tři analogové vstupy, napájecí napětí a společný zemní vodič.

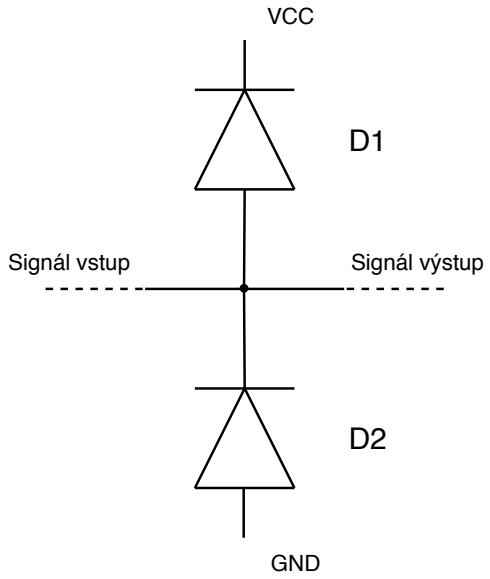
### Ochranné prvky

Vzhledem k tomu, že většina pinů STM32 může být připojena k externí aplikaci, je nutné obvod ochránit před možností poškození. Je třeba počítat s několika možnými vlivy.

První problematický vliv je zkrat skrze STM32. Ten může nastat, pokud je výstupní pin konfigurován jako logická 1 a připojen přímo na společný zemní vodič

(Opačný případ je pin konfigurován jako logická 0 přímo propojen s napájením). Tímto by mohlo dojít k úplnému zničení příslušného pinu, brány nebo dokonce celého mikrokontroléru. Jako ochrana je připojen rezistor v sérii s hodnotou 300 Ω, který omezí zkratový proud na bezpečné hodnoty.

Druhý velmi problematický vliv je indukce nechtěného napětí především při použití dlouhého vedení signálu. Tento vliv je potlačen tzv. clamping diodami, tedy diodami zapojenými do série v závěrném směru. Toto zapojení lze vidět na obrázku 6.4. Dioda D1 je otevřena v případě, že $U_s > U_{D1} + VCC$. Dioda D2 je otevřena v případě $U_s < -U_{D2}$. Obvod chrání z obou stran, rozptyl napětí je dán typem diod, které jsou použity.




Obr. 6.4: Clamping diody schéma zapojení

V neposlední řadě jsou napájecí vstupy vybaveny ochranou proti obrácené polaritě. Oproti klasickému řešení s diodou v propustném směru je k použití zvolen P kanálový MOSFET (Metal Oxide Semiconductor Field Effect Transistor) tranzistor. Využívá se konstrukčního jevu, kdy je mezi vývody Drain a Source dioda v propustném směru. Velkou výhodou použití tranzistoru jsou mnohem menší ztráty oproti diodě, které jsou závislé na odporu tranzistoru v propustném směru.

## 6.3 Firmware

Celý firmware je napsaný v programovacím jazyce C s pomocí knihovny LibOpenCM3 [30]. Knihovna obsahuje velmi základní abstrakce k ovládání mikrokon-

troléru a jeho periferií. Firmware je rozdělen do několika modulů, celý zdrojový kód je k dispozici v adresáři *Node*. První část, která se nachází v podadresáři *lib/periph* obsahuje funkce a definice pro ovládání jednotlivých periferií. K dispozici jsou následující moduly:

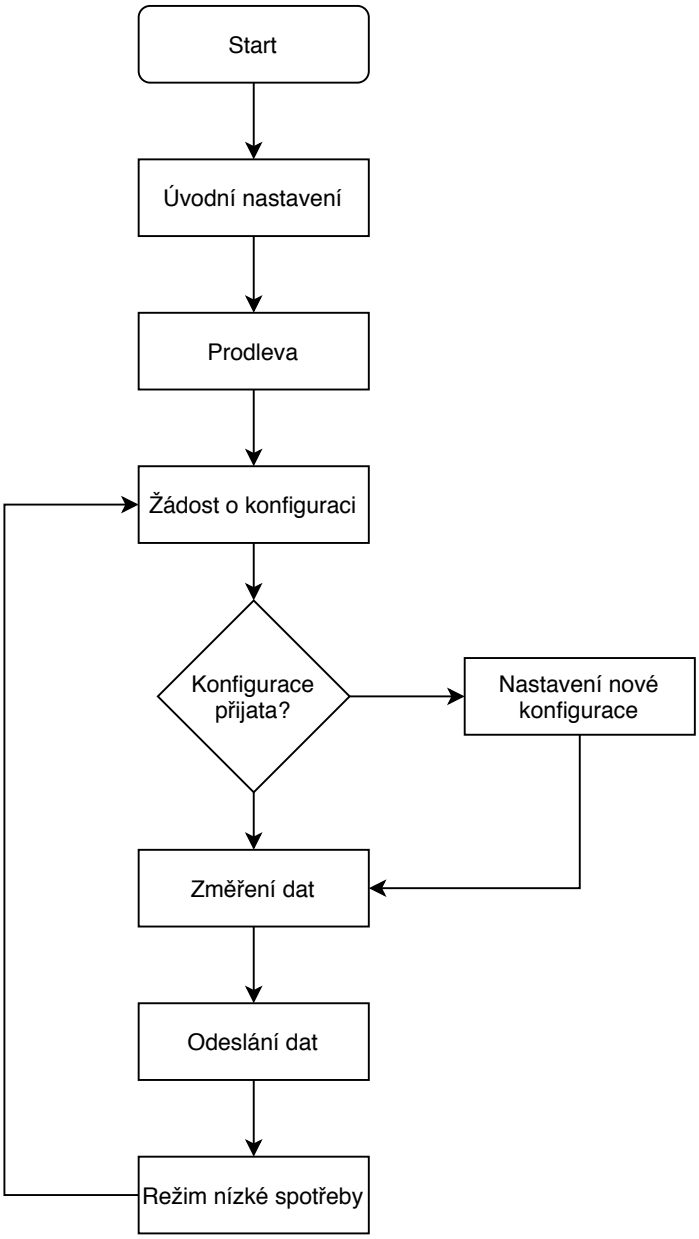
- adc - ADC
- gpio -GPIO piny
- i2c - I2C sběrnice
- low_power - Režim nízké spotřeby
- rtc - Obvod reálného času
- spi - SPI sběrnice
- sys_tick - Systémový časovač
- usart - UART sběrnice

K ovládání bezdrátového přenosu a měření s pomocí BME280 slouží moduly *rfm69* a *bme280*. Poslední modul je vlastní program *main* v podadresáři *node*. Hlavní část programu je zobrazena v diagramu 6.5.

Hned po startu programu probíhá nastavení všech periferií:

- Zdroj hodinového signálu - Je nastaven vnitřní vysokorychlostní oscilátor s frekvencí 16 MHz. Zároveň je v tomto kroku zpřístupněn hodinový signál pro použité periferie.
- GPIO piny pro periferie - Za zmínku stojí především nastavení SPI, kde je z důvodu HW chyby mikrokontroléru, potřeba povolit vysokou rychlost pro komunikační piny. Na druhou stranu u I2C je třeba nastavit piny jako open-drain konfiguraci.
- Systémový časovač - Čas obnovení je nastaven na 1 ms, zpožďovací funkce má tedy nejmenší možný interval také 1 ms.
- SPI sběrnice
- Obvod reálného času - Jako zdroj hodinového signálu je nastaven vnitřní nízkorychlostní oscilátor s frekvencí 37 kHz.
- UART sběrnice
- RFM69 bezdrátový modul
- ADC - Pro zvýšení přesnosti měření je nabíjecí čas převodníku stanoven na 39,5 cyklu.
- I2C sběrnice
- Senzor BME280
- Uživatelské GPIO piny

Vzhledem k tomu, že zařízení tráví většinu času v režimu nízké spotřeby, bylo třeba rozhodnout jakým způsobem bude doručena nová konfigurace do zařízení. Ke spotřebě šetrný způsob je žádost o konfiguraci hned po probuzení. Zařízení tak nemusí stále poslouchat a čekat na příjem. Nevýhodou je, že se konfigurace promítne




Obr. 6.5: Vývojový diagram hlavní části programu

po nejbližším probuzení. V krajním případě to může být doba celého režimu nízké spotřeby. Na paket se čeká 1 s, pokud ve stanoveném čase nedorazí odezva, je jasné, že není žádná nová konfigurace.

Měření dat probíhá postupně dle možností softwaru. Nejprve je vyžádán vzorek ze senzoru BME280, jehož operace probíhá asynchronně. Dále je zapnut tranzistor pro měření napětí baterie a jsou posbírány relevantní vzorky z ADC kanálů. Poté dojde ke sběru dat z uživatelských GPIO pinů. Vše je zapsáno do paketu. V posledním kroku se čeká na konec měření BME280. Jsou vypočítány hodnoty teploty,

tlaku, vlhkosti a přidány do paketu.

Data jsou následně odeslána s nutností potvrzení, na které se čeká 100 ms. V případě, že žádné nedorazí, jsou data odeslána znovu. Toto je opakováno až pětkrát.

Nakonec zařízení přechází do režimu nízké spotřeby. Jsou vypnuty všechny periferie s výjimkou obvodu reálného času, který je schopný probudit mikrokontrolér po uplynutém čase. RFM69 a BME280 jsou také uvedeny do režimu nízké spotřeby k ušetření co největšího množství energie.

### Kompilace

Důležitou součástí vývoje SW pro mikrokontroléry jsou nástroje pro kompilaci. Velkou výhodou je existence *GNU Arm Embedded Toolchain*, který je schopen zkompilovat kód pro většinu ARM procesorů z rodiny Cortex-M a Cortex-R. Avšak instalace potřebných nástrojů může být občas zdlouhavá, a proto byl vytvořen kompilační kontejner. Ten obsahuje vše potřebné ke kompilaci. Jediná nutnost je tedy použití programu *Docker* nebo *Podman*. Před prvním použitím je kontejner nutné sestavit. Z kořenového adresáře stačí spustit příkaz A.1. Takto vytvořený kontejner může být použit pokaždé, je tedy nutno jej vytvořit pouze jednou. Samotná kompilace firmwaru je vykonána za použití příkazu A.2. Výsledkem jsou dva soubory *node.elf* a *node.bin*. Soubor s příponou *elf* je určen ke krokování programu a obsahuje všechny potřebné informace.

# 7 Brána

Téměř jediný požadavek na bránu je možnost připojení do sítě Internet. Toho lze velmi jednoduše dosáhnout za použití HW se síťovou kartou, na který lze nainstalovat operační systém.

## 7.1 Výběr HW

Z důvodů uvedených výše byl vybrán jednodeskový počítač Raspberry Pi, který je velice populární pro různá řešení a domácí automatizace. Pro potřeby brány byl vybrán model Raspberry Pi 4 s 2 GB operační paměti. Jedná se o poslední model s větším výkonem oproti předcházejícím modelům. Cena tohoto modelu je přibližně 1000 Kč bez příslušenství. Nezbytné příslušenství pro fungování celého počítače je paměťová karta a napájecí zdroj.

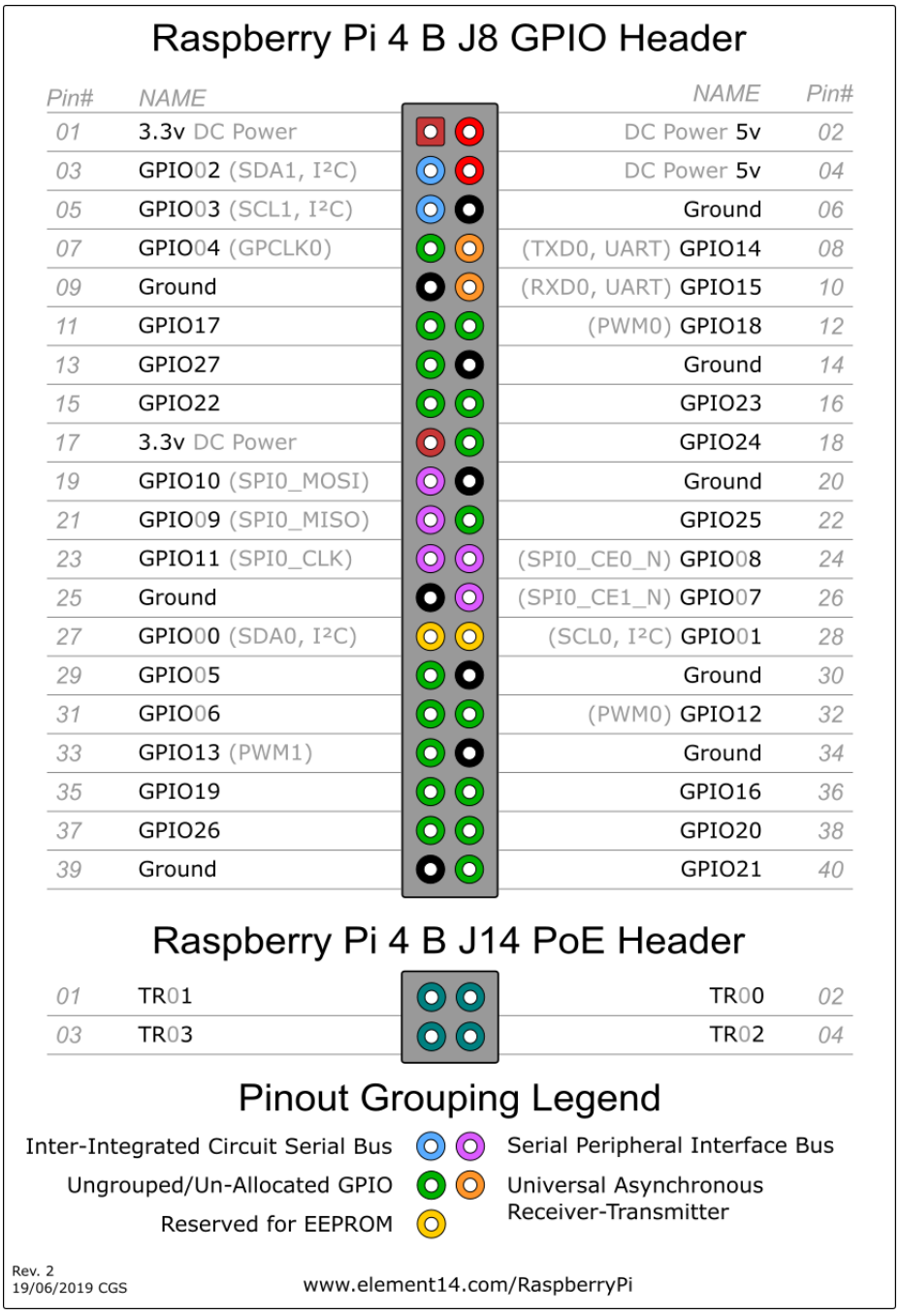
### Technické parametry

- Procesor - 64 bitový ARM Cortex-A72 s maximální frekvencí 1,5 GHz
- Paměť - LPDDR4 (Low-Power Double Data Rate 4) s kapacitou 2GB
- Síťová konektivita - 2,4 GHz a 5 GHz IEEE 802.11.b/g/n/ac Wi-Fi, Gigabit Ethernet
- Bluetooth 5.0
- USB - Dva porty verze 2.0 a dva verze 3.0
- GPIO - 40 pinový GPIO header, viz 7.1
- Video výstup - Dva microHDMI konektory
- Napájení - 5 V přes USB-C nebo GPIO header, minimálně 3 A

### Operační systém

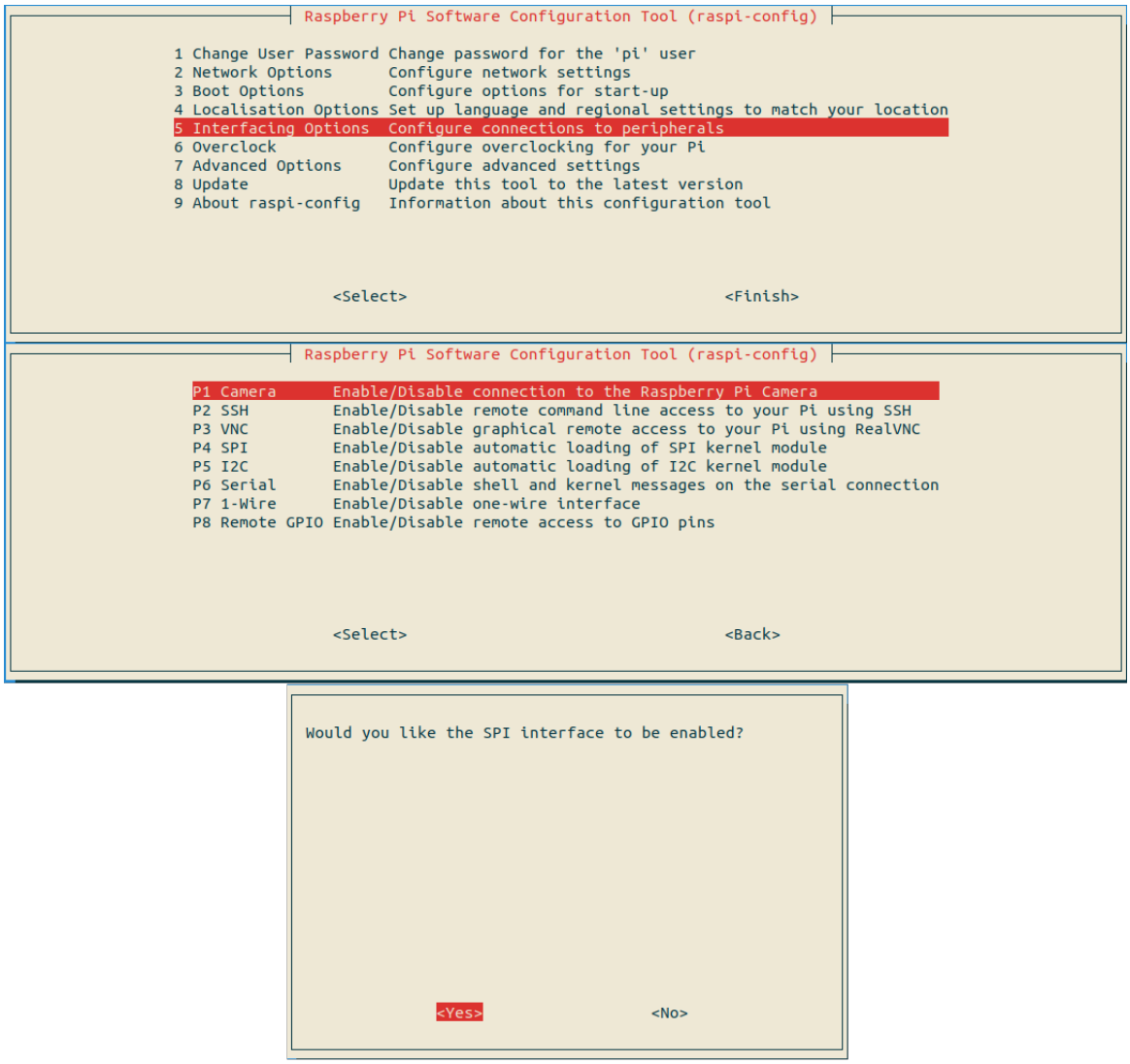
Pro potřeby brány byl zvolen operační systém Raspbian Buster Lite, který je založen na Linuxové distribuci Debain. Nejnovější verze je zdarma ke stažení na stránkách `https://www.raspberrypi.org/downloads/`. Výsledný soubor je zabalen ve formátu *zip*. Po rozbalení je třeba soubor s příponou *.img* zapsat na zformátovanou SD kartu. Zapsání lze provést několika programy avšak v tomto případě byl zvolen program *dd* pomocí příkazu A.3. Je nutné si dát pozor na jméno oddílu, na který probíhá zápis, jinak může dojít k nechtěnému přepsání jiných dat.

Verze Lite neobsahuje žádné grafické rozhraní, a proto je nutné před prvním připojením povolit SSH (Secure Shell) rozhraní, které je z bezpečnostních důvodů vypnuté po instalaci. Pro zapnutí je třeba vytvořit prázdný soubor s názvem *ssh* na */boot* oddílu SD karty. Po zapnutí počítače a nalezení IP adresy se lze připojit.




Obr. 7.1: Raspberry Pi GPIO header [31]

Aby brána mohla komunikovat s RFM69 je třeba také povolit SPI sběrnici v operačním systému. K tomu slouží program *raspi-config* a je nutné jej otevřít s právy *root* uživatele. V tomto okně vybereme možnost s číslem 5 *Interfacing Options*. V dalším menu se jedná o položku P4 *SPI*. Nakonec je volbu nutno potvrdit pomocí *Yes*. Po restartu počítače by měl být k dispozici ovladač a v adresáři */dev/* by se měla nacházet zařízení *spidev0.0* a *spidev0.1*.




Obr. 7.2: Raspberry Pi konfigurace

## 7.2 Software

Úkolem proxy je v podstatě přeložit data přicházející z vzdálené jednotky a poslat je pomocí MQTT do běžícího Home Assistanta. Z tohoto důvodu je proxy program rozdělen na čtyři asynchronně běžící části. Proxy je napsán v programovacím jazyce Rust. Program mimo jiné podporuje možnost konfigurace vzdálené jednotky pomocí konfiguračního souboru. Všechny zdrojové kódy se nachází v adresáři *Gateway/proxy*.

### Studentská spolupráce

V rámci studentské spolupráce jsou data, která se odesílají do Home Assistant také odesílána na MQTT server VUT. Data na tomto serveru jsou využita pro *Sběr telemetrických dat a jejich vyhodnocení* [32]. Data jsou odesílána ve formátu JSON jako pole hodnot na předem stanovený topic *vutbr/xmusil/223202*. Formát jedné hodnoty lze vidět ve výpise 7.1.

Výpis 7.1: JSON hodnota pro studentskou spolupráci

```
{"name":"Digital0","value":"0"}
```

### Knihovna pro RFM69

Jak již bylo zmíněno, program byl napsán v programovacím jazyce Rust. To může být mírně problematické s ohledem na možné mezery v dostupných knihovnách. V tomto případě se jednalo o knihovnu na komunikaci s RFM69. Jediná dostupná knihovna byla velmi zastaralá, a proto byla v rámci práce vytvořena nová funkční knihovna. Zdrojové kódy knihovny jsou dostupné na adrese `https://github.com/almusil/rfm69/`. Kód v knihovně je formován tak, aby bylo možné RFM69 použít v mnohem širším kontextu, než pouze pro komunikaci v rámci práce.

### Běh programu

Běh programu brány je mírně složitější, protože je třeba, aby brána byla schopná vykonávat několik věcí zároveň. Po spuštění programu je načtena konfigurace pro vzdálenou jednotku ze souboru. S ohledem na tuto okolnost je třeba program restartovat po změně konfigurace. Následně proběhne inicializace obou MQTT klientů a do Home Assistant je odeslán discovery dle konfigurace vzdálené jednotky. Dále je provedena inicializace RFM69 s parametry potřebnými pro komunikaci se vzdálenou jednotkou.

Jak již bylo zmíněno, důležité části programu běží asynchronně. První běžící část pouze hlídá signál, tzv. SIGINT nebo SIGTERM, který dorazí v případě přerušení běhu programu. Tato část je nezbytná pro nenásilné ukončení a především k tomu, aby bylo vše uvedeno do stavu jaký byl před spuštěním programu. Vlákno (nejedná se o vlákno v pravém slova smyslu, ale pro potřeby práce to nehraje roli.) je propojeno s hlavním vláknem pomocí komunikačního kanálu.

Druhá část se stará o bezdrátové spojení. Všechna přijatá data jsou dekódovana a předána ke zpracování. Pokud dorazí žádost o konfiguraci, vlákno zkontroluje současný stav konfigurace a, pokud je to potřeba, odešle novou. Pokud se jedná o datový paket, jsou data předána k dalšímu zpracování do hlavního vlákna pomocí komunikačního kanálu.

Ve třetí části běží MQTT klient pro komunikaci s Home Assistant instancí. Opět je k dispozici kanál, který posílá zprávy obdržené z Home Assistant do hlavního vlákna. Samozřejmě je třeba skrz klienta poslat naměřená data.

Z toho plyne operace hlavního vlákna. Hlavní smyčku lze vidět ve výpisu 7.2. Pokud dorazila data ze vzdálené jednotky, jsou předána na zpracování a poslána přes MQTT. Data, která jsou obdržena z Home Assistant, jsou promítnuta v konfiguraci vzdálené jednotky. Poslední blok jsou data signálu pro ukončení hlavní smyčky a celého programu.

Výpis 7.2: Brána hlavní smyčka

```
loop {
  select! {
    opt = receiver.next().fuse() => match opt {
      Some(buffer) => {
        let data = Data::try_from(&buffer[..])?;
        self.mqtt.update_state(&data).await?;
        self.vutbr.update_state(&data).await?;
      },
      None => error!("Radio channel is closed"),
    },
    option = mqtt_receiver.next().fuse() => {
      helper_mqtt_config(self.conf.clone(), option).await?
    },
    sig =  shutdown.next().fuse() => match sig {
      Some(_) => break,
      None => error!("Shutdown channel is closed, should not
 happen")
    },
  }
}
```

### Konfigurace

Vzdálenou jednotku lze konfigurovat pomocí souboru s příponou *config.yaml*, který se nachází v adresáři *conf*. Soubor je ve formátu YAML, který je velmi vhodný především pro konfiguraci. Příklad konfigurace lze vidět ve výpisu 7.3. Konfigurace je rozdělena na sekce:

- *gateway_addr* - Adresa brány (0 - 255)
- *network_id* - ID sítě (0 - 255)

- *node* - Nastavení vzdálené jednotky

  *sleep_time* - Čas strávený v režimu nízké spotřeby, shodný s časem obnovy dat (0 - 65 535)

  *node_addr* - Adresa vzdálené jednotky (0 - 255)

  *digital* - Nastavení digitálních pinů

    D2 - Jméno pinu, které bude zobrazeno v Home Assistantu

    *type* - Určení zda-li bude pin vstupní nebo výstupní (Input, Output)

    *number* - Číslo pinu (0 - 7)

    *state* - Stav pinu po inicializaci (true, false). Stav lze nastavit pouze pro výstupní pin

  *analog* - Nastavení analogových pinů

    A0 - Jméno pinu, které bude zobrazeno v Home Assistantu

    *number* - Číslo pinu (0 - 2)

    *enabled* - Povolení měření (true, false)

    *unit* - Jednotka naměřené hodnoty

    *expr* - Výraz, na základě kterého bude vypočítána výsledná hodnota z naměřené

Výpis 7.3: Příklad konfigurace

```
---
gateway_addr: 1
network_id: 100
node:
  sleep_time: 10
  node_addr: 10
  digital:
    D2:
      type: Input
      number: 2
    D3:
      type: Output
      number: 3
      state: false
  analog:
    A0:
      number: 0
      enabled: true
      unit: "V"
      expr: "{{ (float(value) * 3.3 / (2**12 - 1)) | round(3) }}"
```

### Kompilace

Obdobně jako u firmwaru pro vzdálenou jednotku je kompilace provedena pomocí kontejneru, který obsahuje výsledný program a lze jej rovnou spustit pomocí programu Docker nebo Podman. Kontejner lze vytvořit za použití příkazu A.4 z kořenového adresáře projektu. Za zmínku stojí přizpůsobení kontejneru tak, aby nedocházelo k neustálé kompilaci všech závislostí projektu. Kontejner je rozdělen na vrstvy tak, aby došlo k uložení závislostí v jedné vrstvě a samotný program v jiné.

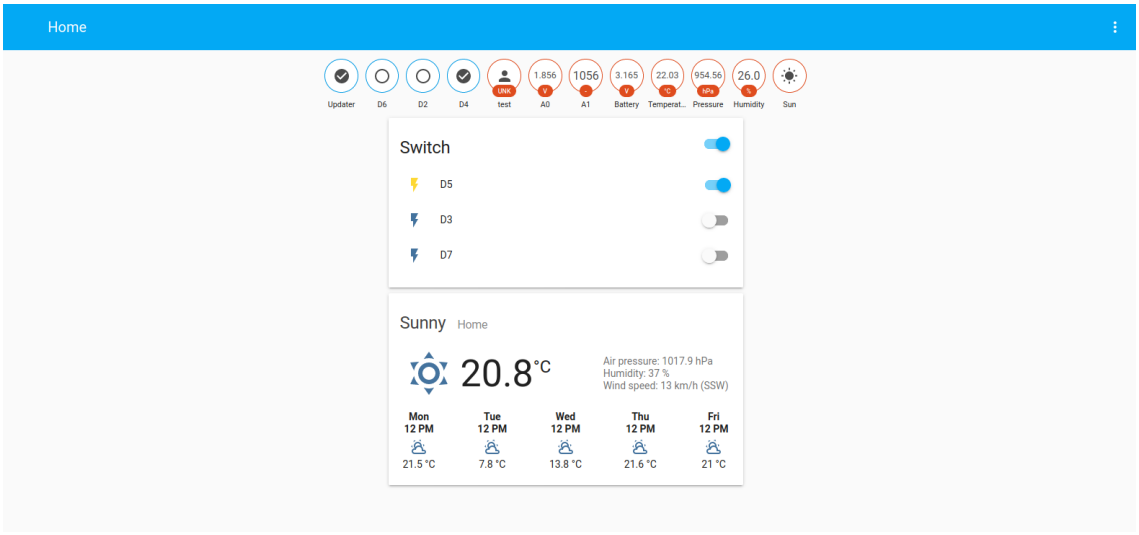
### Spuštění

Celý software brány je složen ze tří komponent - proxy programu, Home Assistant a MQTT serveru Mosquitto. Všechny tři komponenty lze spustit pomocí příkazu A.5 z kořenového adresáře. Před spuštěním je třeba specifikovat proměnnou prostředí s názvem *VUTBR_URI* obsahující URI (Uniform Resource Identifier) MQTT serveru pro studentskou spolupráci. Po spuštění příkazu budou spuštěny všechny prvky

a připojeny do stejnojmenného síťového prostoru.

## 7.3 Home Assistant

Slouží v podstatě jako databáze pro naměřené hodnoty a ovladač s grafickým rozhraním. Po prvním spuštění je třeba zaregistrovat účet administrátora a dále v konfiguraci nastavit MQTT server. K tomu je nutné v adresáři *Gateway/home-assistant/conf* do souboru *configuration.yaml* (Soubor se vytvoří až po prvním spuštění.) přidat výpis A.6. Po restartu se Home Assistant připojí k MQTT serveru a umožní tak automatickou konfiguraci zařízení. Home Assistant obsahuje přehlednou úvodní stránku, na které jsou ukázána všechna momentálně připojená zařízení. Ukázku lze vidět na obrázku 7.3. Je k dispozici také historie s grafy, jak se měnily hodnoty zařízení v čase, viz 7.4.




Obr. 7.3: Home Assistant úvodní strana

### Automatická konfigurace (Discovery)

Home Assistant umožňuje konfigurovat určité druhy zařízení za běhu pomocí MQTT serveru. Tato metoda představuje velkou výhodu, protože zařízení, se kterými Home Assistant pracuje, musejí být zapsána v konfiguračním souboru. Jedinou podmínkou je povolení této služby pro zvolený MQTT server. Je k tomu zapotřebí pouze odeslat konfiguraci ve formátu JSON na určitý topic. Formát topicu je následovný: *home-assistant/{typ}/{jmeno}/config*. Typ zařízení by měl být jeden z podporovaných, v případě brány jsou potřeba pouze tři typy:

- switch - Přepínač


Obr. 7.4: Home Assistant historie

- binary_sensor - Binární senzor
- sensor - Senzor

Jméno by mělo být unikátní, a to z důvodu odstranění konfigurace zařízení. To lze provést tak, že za použití stejného topicu pošleme jen prázdný řetězec místo JSON konfigurace. Formát konfigurace se odlišuje dle zvoleného typu zařízení, ukázku senzoru baterie lze vidět ve výpisu 7.4.

Výpis 7.4: Ukázka dicovery konfigurace senzoru baterie

```
{
  "name":"Battery",
  "state_topic":"node/analog/bat/state",
  "unit_of_measurement":"V",
  "value_template":"{{ ((float(value) * 3.3 / (2**12 - 1)) / 0.8) | round(3) }}"
}
```

# 8 Naměřené hodnoty

### Životnost baterie

Po sestavení vzdálené jednotky byly naměřeny hodnoty odběru proudu během aktivního měření, odesílání dat a v režimu nízké spotřeby. Z těchto hodnot je vypočítána teoretická životnost baterie pro různé intervaly odesílání. V běhu odebírá vzdálená jednotka 25,7 mA, avšak je nutné podotknout, že tato hodnota může být ovlivněna, především zařízením, které odebírá proud ze vstupně výstupních pinů. Odběr v režimu nízké spotřeby je 50 $\mu$A. Stejně jako předchozí hodnota může být závislá na připojeném zařízení. Pro výpočet byla použita kapacita baterie 1200 mAh.

$$\mathrm{I_{active}} = 25,7\ mA = 0,0257\ A$$
$$\mathrm{I_{sleep}} = 50\ \mu A\ = 0,00005\ A$$
$$\mathrm{Kap} = 1200\ mAh = 1,2\ Ah$$
$$\Delta\mathrm{T} = 10\ min = 600\ s$$
$$\mathrm{T_{active}} = 2,5\ s$$
$$\mathrm{T_{sleep}} = \Delta\mathrm{T} - \mathrm{T_{active}}$$
$$\mathrm{T_{sleep}} = 600 - 2,5 = 597,5\ s$$
$$\mathrm{I_{avg}} = (\mathrm{T_{sleep}} * \mathrm{I_{sleep}} + const\mathrm{T_{active}} * \mathrm{I_{active}})/\Delta\mathrm{T}$$
$$\mathrm{I_{avg}} = (597,5 * 0,00005 + 1 * 0,0257)/600 = 0,05565/600 = 0,00009275\ A$$
$$\mathrm{T_{bat}} = \mathrm{Kap}/\mathrm{I_{avg}}$$
$$\mathrm{T_{bat}} = 1,2/0,00009275 = 7649,40\ h \simeq 319\ dní$$

Z výpočtu lze vidět, že při intervalu odesílání dat 10 min je baterie schopná vydržet přibližně 319 dní. Je důležité mít na paměti, že výpočet počítá pouze s použitím baterie a nikoliv se solárním panelem, který životnost značně prodlouží. Tabulka 8.1 uvádí předpokládanou životnost pro různé intervaly měření.

### Dosah rádiové komunikace

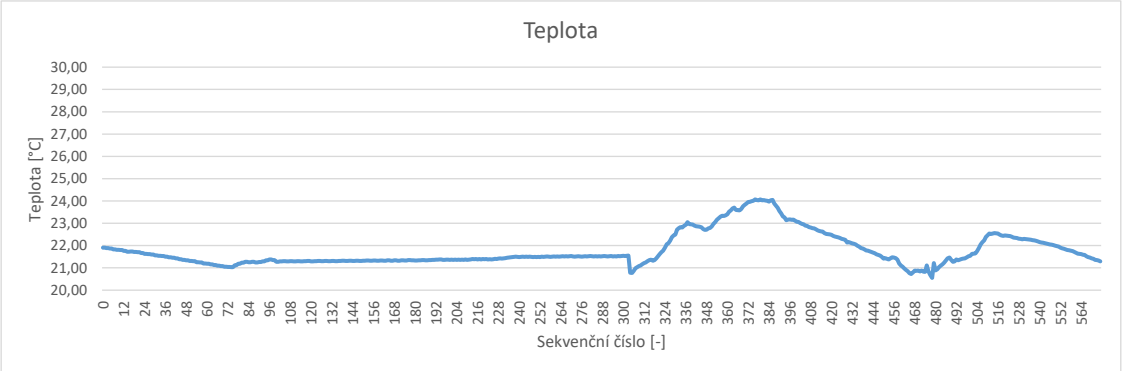
Dále byl změřen teoretický dosah zařízení. V tomto ohledu vzdálená jednotka nedosahuje dobrých výsledků. Bez přidané antény byla komunikace velmi špatná i na malou vzdálenost. Vzdálená jednotka obsahuje místo na DPS pro UFL konektor, který z důvodu omezené dostupnosti nemohl být použit. S drátovou anténou o délce čtvrtiny vlnové délky je vysílač schopný komunikovat přibližně na vzdálenost 6 m. Naměřené hodnoty síly signálu na přijímači lze vidět v tabulkách C.1, C.2 a 8.2. Modul RFM69, který je použit mimo DPS venkovní jednotky, je schopen komunikovat na vzdálenost přibližně 25 m i s několika zdmi v cestě. Špatná přenos signálu je s velkou pravděpodobností zapříčiněn cestou na DPS, která sama působí jako špatná anténa a vytváří parazitní kapacitu vzhledem ke společnému vodiči.

Tab. 8.1: Životnost baterie

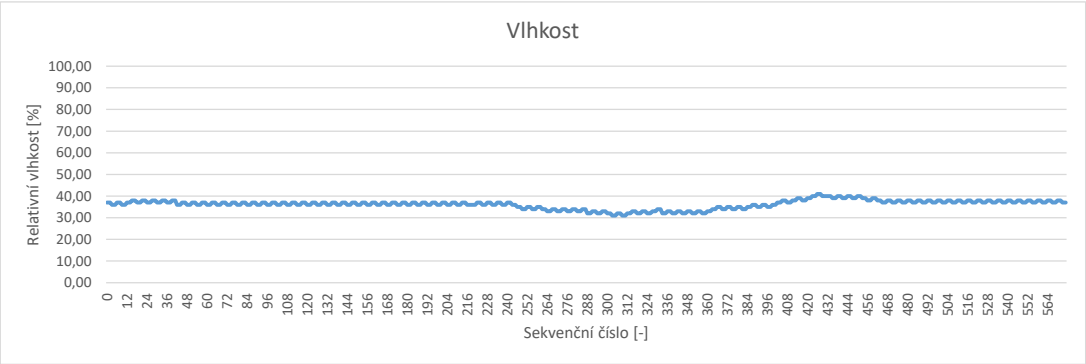
| Interval vysílání | Průměrný proud | Životnost | |
|---|---|---|---|
| min | mA | hodiny | dny |
| 10 | 0,157 | 7 649 | 319 |
| 15 | 0,121 | 9 896 | 412 |
| 20 | 0,103 | 11 601 | 483 |
| 30 | 0,086 | 14 014 | 584 |
| 40 | 0,077 | 15 641 | 652 |
| 50 | 0,071 | 16 812 | 701 |
| 60 | 0,068 | 17 695 | 737 |

### Demonstrační měření

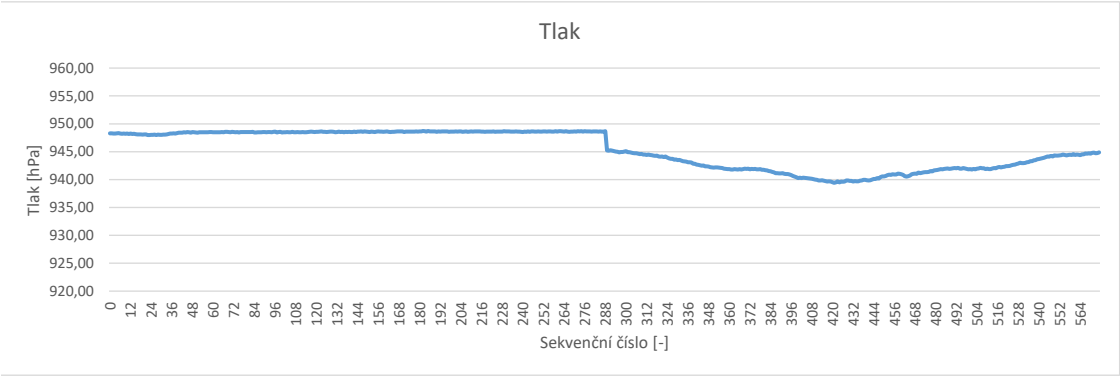
V rámci práce prováděla vzdálená jednotka měření teploty, atmosferického tlaku a relativní vlhkosti vzduchu. Sběr dat probíhal s přestávkami více než dva měsíce. Všechna data byla posílána do Home Assistanta. Přehled dat v historii lze vidět na obrázku 7.4. V následujících grafech lze vidět naměřená data v rozsahu pěti dnů v pokoji uvnitř domu. Atmosferický tlak není přepočítán na hodnotu hladiny moře, ale je zobrazena hodnota naměřená v nadmořské výšce přibližně 550 m.n.m. Graf teploty se nachází na obrázku 8.1, graf relativní vlhkosti vzduchu na obrázku 8.2 a graf atmosferického tlaku na obrázku 8.3.




Obr. 8.1: Graf teploty




Obr. 8.2: Graf relativní vlhkosti




Obr. 8.3: Graf atmosferického tlaku

Tab. 8.2: Síla signálu s anténou, 6 m

| Sekvenční číslo | Síla signálu |
|---|---|
| - | dBm |
| 1 | -78 |
| 2 | -79.5 |
| 3 | -78 |
| 4 | -77 |
| 5 | -80 |
| 6 | -79 |
| 7 | -79 |
| 8 | -78 |
| 9 | -80 |
| 10 | -79.5 |
| 11 | -79 |
| 12 | -79 |
| 13 | -78.5 |
| 14 | -78.5 |
| 15 | -79.5 |
| 16 | -77.5 |
| 17 | -77 |
| 18 | -78.5 |
| 19 | -79 |
| 20 | -80.5 |
| 21 | -81 |
| 22 | -80 |
| 23 | -79 |
| 24 | -81 |

# Závěr

Cílem práce bylo vytvořit bezdrátový měřicí systém s bateriovým napájením. Všechny kroky zadání byly splněny a výsledkem je funkční vzdálená jednotka, která je schopná komunikovat s bránou. Pro vzdálenou jednotku byla vytvořena DPS, na kterou byl osazen vybraný mikrokontrolér STM32L031K6. Deska komunikuje s bránou pomocí RFM69 a má dosah přibližně 6 m, který není ideální. Takto nízký dosah je způsoben chybným návrhem cesty antény u vzdálené jednotky. Způsob vedení a šířka cesty do značné míry ovlivnila parazitní kapacitu, která má velký vliv na výslednou impedanci antény a není tedy možné vysílat s plným výkonem. K napájení vzdálené jednotky slouží LTC3106, který umožňuje využít dva zdroje s plynulým přepínáním. Tohoto faktu bylo využito a vzdálenou jednotku je možné napájet solárním panelem. V této konfiguraci slouží baterie pouze jako záložní zdroj v době, kdy není solární panel schopný dodat dostatečný výkon. Spotřeba vzdálené jednotky se pohybuje okolo 25,7 mA během vysílání, 50 $\mu$A v režimu nízké spotřeby. To poskytuje dobrou životnost i bez použití solárního panelu. Pro baterii o kapacitě 1200 mAh a intervalu vysílání 10 minut se jedná přibližně o 319 dní.

Software pro bránu běží na operačním systému Linux, konkrétně Debian distribuci upravené pro Raspberry Pi 4. V rámci práce byl vytvořen program v programovacím jazyce Rust, který se stará o komunikaci se vzdálenou jednotkou, dále posílá zpracovaná data do instance Home Assistanta pomocí protokolu MQTT. Byla vytvořena knihovna pro Rust umožňující komunikaci za pomoci RFM69. Všechny programy pro bránu jsou dostupné jako kontejnery a jejich výhodou je tedy možnost spustit je i na jiné platformě bez nutnosti instalace dodatečných programů. V rámci studentské spolupráce byla naměřená ukázková data odesílána na MQTT server VUT pro zpracování jinou diplomovou prací.

Měřící systém jako celek lze využít například pro monitorování podmínek v různých uzavřených nebo i venkovních prostorech. Dobrý příklad může být propojení vzdálené brány se systémem pro ovládání oken a dveří. Home Assistant nabízí možnost reakce na určitou událost. Je tedy možné sepnout výstup při překročení určité teploty a zase jej rozepnout pokud teplota klesne.

# Literatura


[1] CHAUDHURI, Abhik. *Internet of Things, for Things, and by Things.* 1. Abingdon-on-Thames, Velká Británie: Routledge, 2019, 1. DOI: 10.1201/9781315200644. ISBN 9781138710443.

[2] Vendor Lock-in Definition. *The Linux Information Project* [online], 29.04.2006 [cit. 2020-02-14]. Dostupné z: <`http://www.linfo.org/vendor_lockin.html`>

[3] YUAN, Fei a Krzysztof INIEWSKI. *Low-Power Circuits for Emerging Applications in Communications, Computing, and Sensing.* 1. Boca Raton: CRC Press, 2018. DOI: 10.1201/9780429507564. ISBN 9781138580015.

[4] CHEN, Cailian, Shanying ZHU, Xinping GUAN a Xuemin (Sherman) SHEN. *Wireless Sensor Networks.* 1. Cham: Springer International Publishing, 2014. DOI: 10.1007/978-3-319-12379-0. ISBN 978-3-319-12378-3.

[5] *IEEE Standard for Low-Rate Wireless Networks.* IEEE Std 802.15.4™-2015. New York: IEEE, 2015.

[6] *Zigbee Alliance* [online]. Davis, USA: 2002 [cit. 2020-02-16]. Dostupné z: <`https://zigbeealliance.org/`>

[7] GISLASON, Drew. *Zigbee wireless networking.* 1. Amsterdam ; Boston: Elsevier / Newnes, 2008, xvi, 425 s. : il. ; 24 cm. ISBN 978-0-7506-8597-9.

[8] RFC 4919. *IETF* [online]. 2007 [cit. 2020-02-17]. Dostupné z: <`https://tools.ietf.org/html/rfc4919`>

[9] SHELBY, Zach a Carsten BORMANN. *6LoWPAN: the wireless embedded internet.* Chichester: John Wiley, 2009, xx, 223 s. : il. ISBN 978-0-470-74799-5.

[10] *IEEE Standard for Information technology-Telecommunications and information exchange between systems Local and metropolitan area networks.* IEEE Std 802.11™-2016. New York, 2016.

[11] LABIOD, H, H AFIFI a C DE SANTIS. *Wi-Fi, Bluetooth, ZigBee and WiMAX.* Dordrecht: Springer, 2007, xvi, 316 s. ISBN 978-1-4020-5396-2.

[12] *IEEE Standard for Wireless Personal Area Network.* IEEE Std 802.15.1™-2002. New York: IEEE, 2002.

[13] MORROW, Robert. *Bluetooth operation and use.* New York: McGrawHill, 2002, xviii, 567 s. : il. ISBN 0-07-138779-X.

[14] RFM69 Specification. *HopeRF* [online]. Pingshan, 2018 [cit. 2020-02-21]. Dostupné z: <https://www.hoperf.com/data/upload/portal/20191105/RFM69%20Specification.pdf>

[15] ARM CPU Architecture. *ARM Developer* [online]. Cambridge, 2018 [cit. 2020-02-21]. Dostupné z: <https://developer.arm.com/architectures/cpu-architecture>

[16] YIU, Joseph. *The definitive guide to the ARM CORTEX-M0.* Oxford: Elsevier ;, Newnes, 2011, xix, 529 s. : obr., grafy. ISBN 978-0-12-385477-3.

[17] *ARM Architecture Reference Manual.* 9. Cambridge, 2005.

[18] Architektura mikrořadičů s jádry ARM Cortex-M0 a ARM Cortex-M0+. *Root.cz* [online]. Praha, 2015 [cit. 2020-02-21]. Dostupné z: <https://www.root.cz/clanky/architektura-mikroradicu-s-jadry-arm-cortex-m0-a-arm-cortex-m0>

[19] Frequently Asked Questions. *MQTT.ORG* [online]. 2014 [cit. 2020-02-22]. Dostupné z: <http://mqtt.org/faq>

[20] GASTON C., Hillar. *MQTT Essentials - A Lightweight IoT Protocol.* United Kingdom: Packt Publishing, 2017. ISBN ISBN 978-1-78728-781-5.

[21] Home Assistant [online]. [cit. 2020-02-25]. Dostupné z: <https://www.home-assistant.io>

[22] Architecture. *Developer Docs* [online]. 2020 [cit. 2020-02-28]. Dostupné z: <https://developers.home-assistant.io/docs/architecture_index>

[23] KLABNIK, Steve a Carol NICHOLS. *The Rust Programming Language.* 2018. San Francisco: No Starch Press, 2019. ISBN 9781718500440.

[24] Detail kmitočtového pásma: Pásmo 890 – 942 MHz. *Vyžití rádového spektra* [online]. Praha, 2020 [cit. 2020-03-28]. Dostupné z: <http://spektrum.ctu.cz/kmitocty/890-942mhz>

[25] Detail kmitočtového pásma: Pásmo 862 – 890 MHz. *Vyžití rádového spektra* [online]. Praha, 2020 [cit. 2020-03-28]. Dostupné z: <http://spektrum.ctu.cz/kmitocty/862-890mhz>

[26] Detail kmitočtového pásma: Pásmo 432 – 438 MHz. *Vyžití rádového spektra* [online]. Praha, 2020 [cit. 2020-03-28]. Dostupné z: <http://spektrum.ctu.cz/kmitocty/432-438mhz>

[27] *Všeobecné oprávnění č. VO-R/10/12.2017-10 k využívání rádiových kmitočtů a k provozování zařízení krátkého dosahu.* Praha: Český telekomunikační úřad, 2017.

[28] LTC3106. *Analog Devices* [online]. [cit. 2020-03-31]. Dostupné z: <https://www.analog.com/en/products/ltc3106.html>

[29] STM32L031K6. *STMicroelectronics* [online]. [cit. 2020-04-02]. Dostupné z: <https://www.st.com/en/microcontrollers-microprocessors/stm32l031k6.html>

[30] *LibOpenCM3* [online]. [cit. 2020-04-07]. Dostupné z: <https://libopencm3.org/>

[31] Raspberry Pi 4 Model B Default GPIO Pinout with PoE Header. *Element14* [online]. 23.09.2019 [cit. 2020-04-10]. Dostupné z: <https://www.element14.com/community/docs/DOC-92640/l/raspberry-pi-4-model-b-default-gpio-pinout-with-poe-header>

[32] SPURNÝ, Jan. *Sběr telemetrických dat a jejich vyhodnocení* [online]. Brno, 2020 [cit. 2020-02-20]. Dostupné z: <https://www.vutbr.cz/studenti/zav-prace/detail/126046.> Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Petr Číka.

# Seznam symbolů, veličin a zkratek

**IoT** Internet of Things

**HW** Hardware

**DPS** Deska plošných spojů

**SW** Software

**LR-WPAN** Low-rate wireless personal area networks

**ISO/OSI** International Standards Organization/Open Systems Interconnection

**FFD** Full-function device

**RFD** Reduced-function device

**PHY** PHY

**MAC** MAC

**SAP** Service Access Point

**NWK** NWK

**AES** Advanced Encryption Standard

**APS** APS

**ZDO** ZigBee Device Object

**SoC** System on Chip

**IETF** Internet Engineering Task Force

**IP** Internet Protocol

**UDP** User Datagram Packet

**6LoWPAN-ND** 6LoWPAN Neighbor Discovery

**AP** Access Point

**FHSS** Frequency Hopping Spread Spectrum

**BLE** Bluetooth Low Energy

**SPI** Serial Peripheral Interface

| | |
|---|---|
| **FIFO** | First in First out |
| **CRC** | Cyclic Redundancy Check |
| **MMU** | Memory Managment Unit |
| **ECC** | Error Correction Code |
| **NVIC** | Nested Vectored Interrupt Controller |
| **SP** | Stack pointer |
| **LR** | Link register |
| **PC** | Program counter |
| **SWI** | Software interrupt |
| **IRQ** | Interrupt request |
| **FIQ** | Fast interrupt |
| **MQTT** | Message Queuing Telemetry Transport |
| **QoS** | Quality of Service |
| **TCP** | Transmission Control Protocol |
| **TLS** | Transport Layer Security |
| **GC** | Garbage Collector |
| **ERP** | Equivalent Radiated Power |
| **ACK** | Acknowledgement |
| **ReqACK** | Acknowledgement Request |
| **LiPo** | Lithium Polymer |
| **PLL** | Phase Locked Loop |
| **RTC** | Realtime Clock |
| **ADC** | Analog-Digital Converter |
| **I2C** | Inter-Integrated Circuit |
| **SWD** | Serial Wire Debug |

| | |
|---|---|
| **UART** | Universal Asynchronous Receiver Transmitter |
| **GPIO** | General-Purpose Input/Output |
| **MOSFET** | Metal Oxide Semiconductor Field Effect Transistor |
| **USB** | Universal Serial Bus |
| **LPDDR** | Low-Power Double Data Rate |
| **SSH** | Secure Shell |
| **URI** | Uniform Resource Identifier |

# Seznam příloh

# A Užitečně výpisy

## A.1 Vzdálená jednotka

Výpis A.1: Setavení kontejneru pro kompilaci vzdálené jednotky

```
docker build --rm \
-t weather-station/node-build-image \
-f ./docker/Dockerfile.node \
.
```

Výpis A.2: Setavení firmwaru pro vzdálenou jednotku

```
./scripts/run-build-node.sh
```

## A.2 Brána

Výpis A.3: Zápis Raspbianu na SD kartu

```
sudo dd bs=1M if=raspbian.img of=/dev/sdb
```

Výpis A.4: Setavení kontejneru brány

```
docker build --rm \
  -t weather-station/proxy-image \
  -f ./docker/Dockerfile.proxy \
  .
```

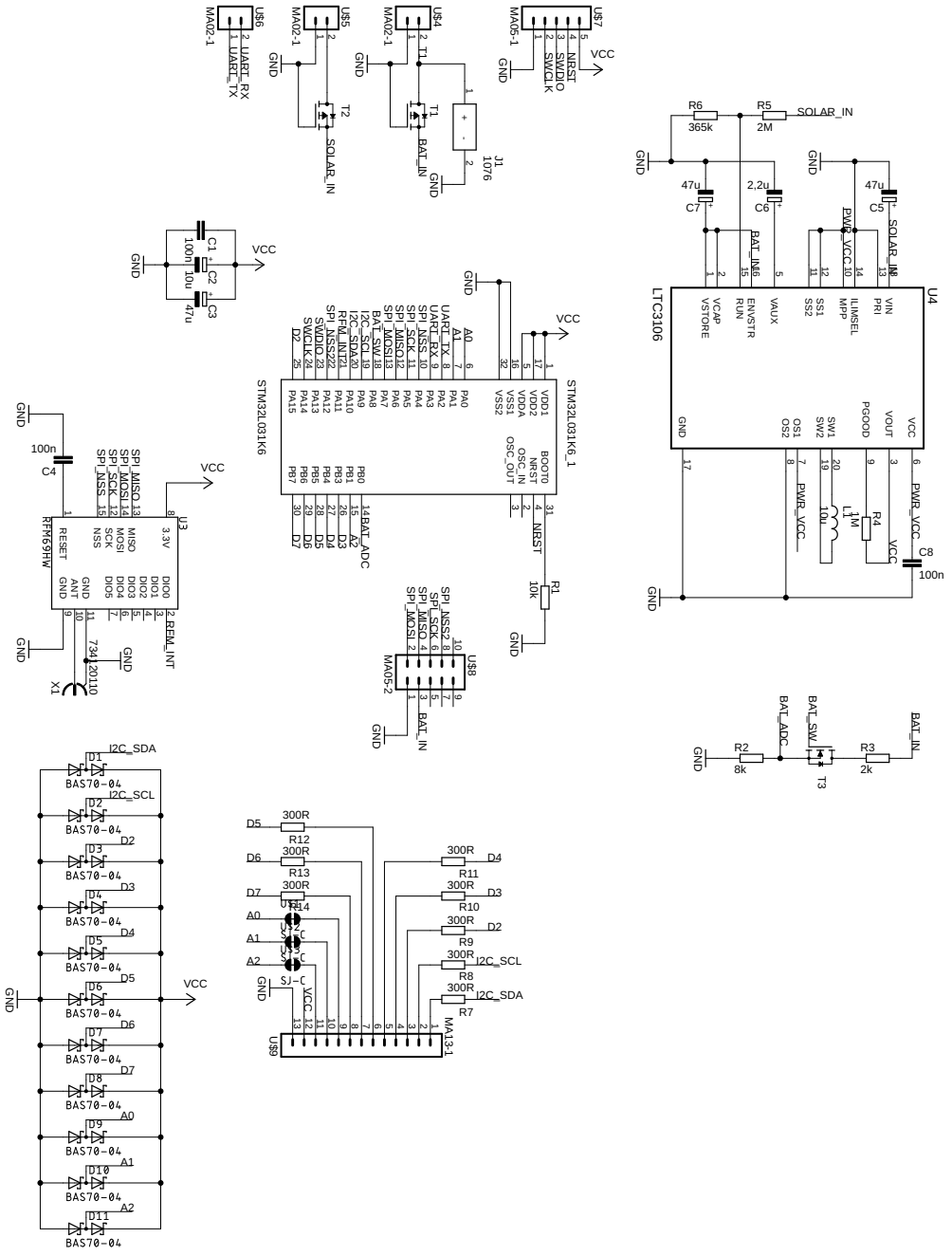
Výpis A.5: Spuštění softwaru pro bránu

```
./scripts/run-gateway-containers.sh
```

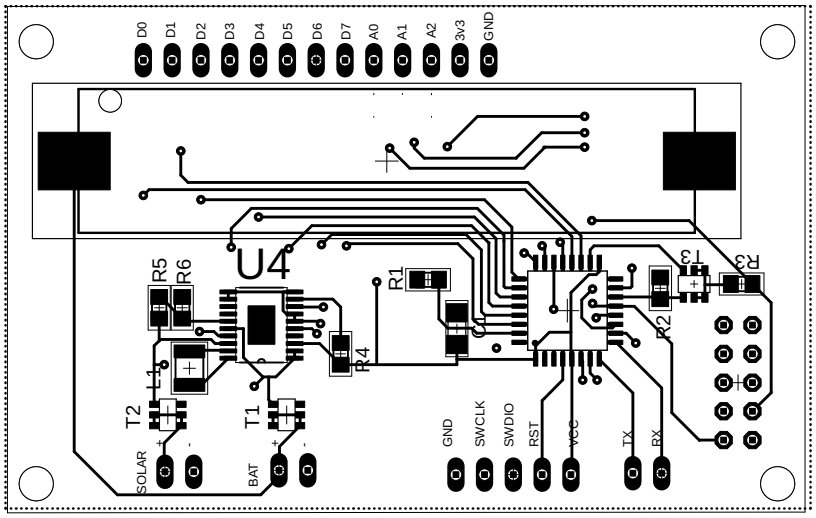
Výpis A.6: Home Assistant MQTT server konfigurace

```
mqtt:
  broker: 127.0.0.1
  port: 1883
  discovery: true
  discovery_prefix: homeassistant
```

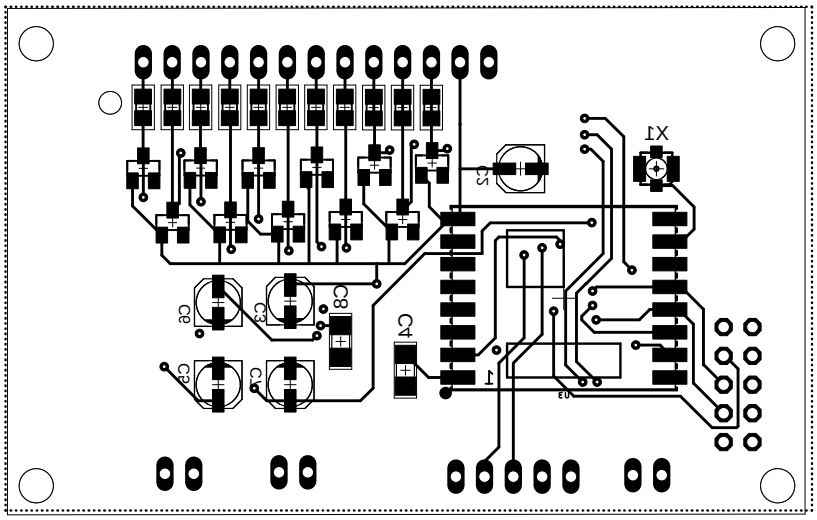
# B Schéma zapojení a DPS



Obr. B.1: Schéma zapojení vzdálené jednotky




Obr. B.2: DPS pohled shora




Obr. B.3: DPS pohled zespodu

# C Naměřené hodnoty

Tab. C.1: Síla signálu bez antény, 10 cm

| Sekvenční číslo | Síla signálu |
|---|---|
| - | dBm |
| 1 | -100 |
| 2 | -100 |
| 3 | -102.5 |
| 4 | -99.5 |
| 5 | -99.5 |
| 6 | -100.5 |
| 7 | -102.5 |
| 8 | -99.5 |
| 9 | -101.5 |
| 10 | -103 |
| 11 | -101 |
| 12 | -102.5 |
| 13 | -100.5 |
| 14 | -99.5 |
| 15 | -101 |
| 16 | -99.5 |
| 17 | -98 |
| 18 | -99.5 |
| 19 | -98 |
| 20 | -102.5 |
| 21 | -99.5 |
| 22 | -100 |
| 23 | -98.5 |
| 24 | -101.5 |

Tab. C.2: Síla signálu bez antény, 2 m

| Sekvenční číslo | Síla signálu |
|---|---|
| - | dBm |
| 1 | -103.5 |
| 2 | -101.5 |
| 3 | -103 |
| 4 | -102 |
| 5 | -107 |
| 6 | -102 |
| 7 | -104 |
| 8 | -105.5 |
| 9 | -99.5 |
| 10 | -102.5 |
| 11 | -106 |
| 12 | -104.5 |
| 13 | -104 |
| 14 | -105.5 |
| 15 | -104.5 |
| 16 | -100.5 |
| 17 | -101 |
| 18 | -105 |
| 19 | -105 |
| 20 | -101.5 |
| 21 | -104 |
| 22 | -99.5 |
| 23 | -101 |
| 24 | -102.5 |

# D Obsah přiloženého CD

- `/` ..... kořenový adresář přiloženého archivu
  - `docker` ..... definice vytvořených kontejnerů
    - `Dockerfile.node`
    - `Dockerfile.proxy`
  - `Gateway` ..... soubory pro programy brány
    - `home-assistant`
      - `conf` ..... uložení souboru Home Assistanta
    - `mosquitto` ..... konfigurace a log MQTT serveru
      - `conf`
        - `mosquitto.conf`
      - `log`
    - `proxy` ..... zdrojové soubory hlavního programu brány
      - `Cargo.lock`
      - `Cargo.toml`
        - `conf`
          - `config.yaml` ..... konfigurace programu brány
        - `log`
        - `src`
          - `config.rs`
          - `data.rs`
          - `error.rs`
          - `home_assistant.rs`
          - `main.rs`
          - `proxy.rs`
          - `radio.rs`
          - `util.rs`
          - `vutbr.rs`
  - `HW`
    - `Node`
      - `with_solar_universal.brd` ..... DPS vzdálené jednotky
      - `with_solar_universal.sch` ..... schéma vzdálené jednotky
  - `LICENSE`

```
/
├── Node ..... zdrojové soubory firmwaru vzdálené jednotky
│   ├── lib
│   │   ├── bme280.c
│   │   ├── include
│   │   │   ├── bme280.h
│   │   │   ├── config.h
│   │   │   ├── periph
│   │   │   │   ├── adc.h
│   │   │   │   ├── gpio.h
│   │   │   │   ├── i2c.h
│   │   │   │   ├── low_power.h
│   │   │   │   ├── rtc.h
│   │   │   │   ├── spi.h
│   │   │   │   ├── sys_tick.h
│   │   │   │   └── usart.h
│   │   │   ├── rfm69_defs.h
│   │   │   └── rfm69.h
│   │   ├── periph
│   │   │   ├── adc.c
│   │   │   ├── gpio.c
│   │   │   ├── i2c.c
│   │   │   ├── low_power.c
│   │   │   ├── rtc.c
│   │   │   ├── spi.c
│   │   │   ├── sys_tick.c
│   │   │   └── usart.c
│   │   └── rfm69.c
│   ├── node
│   │   ├── main.c
│   │   └── Makefile
│   └── rules.mk
├── README.md
├── scripts ..... shellovské skripty
│   ├── run-build-node.sh ..... skript pro kompilaci firmwaru vzdálené jednotky
│   └── run-gateway-containers.sh ..... skript pro spuštění všech programů brány
├── humidity.csv ..... data naměřené vlhkosti vzduchu
├── pressure.csv ..... data naměřeného atmosferického tlaku
├── temperature.csv ..... data naměřené teploty
└── xmusil59.pdf ..... diplomová práce ve formátu pdf
```